POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE
Dipartimento di Elettronica, Informazione e Bioingegneria

Ph.D. in Information Technology

Multimodal Conversational Interfaces: Design, Modelling, Applications

Author:
Pietro Crovari

Supervisor:
Prof. Franca Garzotto

Co-supervisors:
Prof. Stefano Ceri

Ph.d. Thesis
XXXV Cycle

Abstract

A conversational agent is a software that mimics human conversation. They are becoming increasingly successful and adopted in a wide range of domains, such as education, user assistance, mental health, and home automation.

In recent years, the interaction with conversational agents has been blended with other interaction modalities to increase the system's capabilities, creating new multimodal paradigms for interaction. However, this integration is still limited from a methodological perspective despite being broadly exploited.

This Ph.D. research investigates the design, modeling, and development of multimodal conversational agents. This work starts exploring this domain from the design of GeCoAgent and DSBot, two conversational agents to support the data science process. GeCoAgent is a multimodal conversational platform to enable biologists and clinicians to define data analysis pipelines on genomic data through dialogue. The platform automatically translates it into code, executes it, and returns the user the results. GeCoAgent's design process also led to modeling the bioinformatics tertiary analysis process in the form of an ontology that can be used as a reference to elicit the requirements for interactive applications.

DSBot evolves this concept by providing a tool that translates users' research questions, expressed in natural language, into executable pipelines. The system exploits autoML methodologies to select the best algorithm and optimize the parameter selection automatically. Users are involved in the process through the conversation when decisions related to the meaning of the data must be taken. In addition, we release one of the two modules of DSBot as an open-source framework for multimodal conversational troubleshooting.

Having assessed the potentialities of multimodal conversational interfaces, we realize that their design is a largely unexplored field. For this reason, we survey the literature to elicit a set of principles to follow during the design process, and we formalize a conceptual framework to describe the possible degrees of integration of conversational agents and other interfaces.

Then, we complement the finding in the literature by analyzing the impact of multimodality on the conversational experience from a linguistic perspective. We observe users' linguistic performances in a comparative study with more than 120 participants to assess how the introduction of graphical elements affects the conversational experience.

We use these findings to ground the formulation of a conceptual model to support the design process of multimodal conversational interfaces. The model exploits hierarchical schemes, inspired by BPMN formalism, to model conversational interaction and separate the task's description from how it is reified on the various modalities.

In the last part of the thesis, we describe Albot Einstein, a case study of a multimodal pedagogical conversational agent to teach pH to children. In addition, to validate the descriptive capabilities of the model, we test the platform's efficacy in a comparative study with 28 children, obtaining results comparable to the ones achieved through a "traditional" interactive web application.

We design and develop a graphical authoring tool that enables that transform expressed in a notation derived from one of the model into an instance of the application backend. An empirical evaluation with 15 developers shows how such an interface can support developing multimodal conversational interfaces.

Finally, we discuss how the work presented can be framed in a single framework that covers a multimodal conversational agent's whole design and implementation process.

Sommario

Un agente conversazionale è un software che imita la conversazione umana. Questi agenti stanno riscuotendo sempre più successo e sono adottati in un'ampia gamma di settori, come l'istruzione, l'assistenza agli utenti, la salute mentale e la domotica.

Negli ultimi anni, l'interazione con gli agenti conversazionali è stata integrata con altre modalità di interazione per aumentare le capacità del sistema, creando nuovi paradigmi di interazione multimodale. Tuttavia, nonostante sia ampiamente sfruttata, questa integrazione è ancora limitata da un punto di vista metodologico.

Questa tesi studia la progettazione, la modellazione e lo sviluppo di agenti conversazionali multimodali. Il lavoro inizia ad esplorare questo dominio partendo dalla progettazione di GeCoAgent e DSBot, due agenti conversazionali a supporto del processo di data science. GeCoAgent è una piattaforma conversazionale multimodale che consente a biologi e clinici di definire pipeline di analisi dei dati genomici attraverso il dialogo con il sistema. La piattaforma lo traduce automaticamente in codice, lo esegue e restituisce all'utente i risultati. Il processo di progettazione di GeCoAgent ha portato anche a modellare il processo di analisi terziaria bioinformatica sotto forma di un'ontologia che può essere utilizzata come riferimento per elicitare i requisiti delle applicazioni interattive.

DSBot evolve questo concetto fornendo uno strumento che traduce le domande di ricerca degli utenti, espresse in linguaggio naturale, in pipeline eseguibili. Il sistema sfrutta le metodologie autoML per selezionare l'algoritmo migliore e ottimizzare la scelta dei parametri in modo automatico. Gli utenti sono coinvolti nel processo attraverso la conversazione quando devono essere prese decisioni relative al significato dei dati. Inoltre, abbiamo rilasciato uno dei due moduli di DSBot come framework open-source per la risoluzione di problemi conversazionali multimodali.

Dopo aver valutato le potenzialità delle interfacce conversazionali multimodali, ci siamo resi conto che la loro progettazione è un dominio largamente inesplorato. Per questo motivo, abbiamo analizzato la letteratura per elicitare una serie di principi da seguire durante il processo di progettazione e abbiamo formalizzato un quadro concettuale per descrivere i possibili gradi di integrazione tra agenti conversazionali e interfacce che sfruttano altre modalità.

Successivamente, integriamo i risultati della letteratura tramite l'analisi dell'impatto della multimodalità sull'esperienza conversazionale da una prospettiva linguistica. Osserviamo le performance linguistiche degli utenti in uno studio comparativo con più di 200 partecipanti per valutare come l'introduzione di elementi grafici influisca sull'esperienza conversazionale.

Su questi risultati, fondiamo la formulazione di un modello concettuale a supporto del processo di progettazione di interfacce conversazionali multimodali. Il modello sfrutta diagrammi gerarchici, ispirati al formalismo BPMN, per modellare l'interazione conversazionale e separare la descrizione del compito da come viene reificato nelle varie modalità.

Nell'ultima parte della tesi, descriviamo Albot Einstein, un caso di studio di un agente conversazionale pedagogico multimodale per insegnare il pH ai bambini. Inoltre, per validare le capacità descrittive del modello, testiamo l'efficacia della piattaforma in uno studio comparativo con 28 bambini, ottenendo risultati paragonabili a quelli ottenuti con un'applicazione web interattiva "tradizionale".

Abbiamo progettato e sviluppato uno strumento di authoring grafico che consente di trasformare le informazioni espresse in una notazione derivata da quella del modello in un'istanza di backend dell'applicazione. Una valutazione empirica con 15 sviluppatori mostra come tale interfaccia possa supportare lo sviluppo di interfacce conversazionali multimodali.

Infine, discutiamo come il lavoro presentato possa essere inquadrato in un unico framework che copre l'intero processo di progettazione e implementazione di un agente conversazionale multimodale.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation and Goals

A conversational agent is a computer program that mimics human conversation [1]. Users speak or write in natural language (e.g., plain English); the software elaborates the input, extracts the relevant information, formulates a response in natural language, and returns it [2].

In recent years, conversational agents have seen rapid development, both in terms of adoption and technological evolution. Thanks to the advancements in the fields of artificial intelligence, machine learning, and natural language processing, modern conversational agents can interpret increasingly complex requests from users, provide answers on broad domains, and support users with personalized responses keeping in consideration the context of the interaction.

Today, conversational agents are being used successfully employed in many domains, such as customer assistance, e-commerce, education, mental health, and home automation. The presence in society is still at the beginning: Deloitte estimated that the market for conversational AI, starting from $6B value in 2019, will grow to $22.6B by 2024 [3]. The latest innovations in generative conversational agents, i.e., the conversational engines not trained on a set of rules pre-defined by the developer but on a large dataset of existing conversations, are thought to disrupt many online services as they are today, such as search engines [4] and programming environments [5].

Many factors concur with the success of these technologies. First, being the natural language a very intuitive way for humans to interact with technology, the use of conversational agents allows users to relieve the cognitive load of interacting with an application and, therefore, to focus on the task people are facing [6, 7]. Second, this technology offers much room for interaction personalization, therefore providing a better user experience and more comprehensive accessibility of the system [8]. In addition, automatic and self-administered support enabled by conversational agents implies for the user a continuous availability of the service, and for the service provider a lower cost than traditional customer services [9]. Finally, in domains where privacy is particularly relevant (such as in the field of sexually transmitted diseases), the absence of a human respondent is appreciated and is seen as an incentive for users [10].

My Ph.D. journey started by trying to harness the potential of conversational technologies to help people with little or no expertise in computer science and statistics to do data analysis. This gave rise to GeCoAGent, the tool presented in Chapter 2, which evolved into

DSBot, presented in Chapter 3. These two web applications guide users in composing machine learning pipelines through a conversational agent, the first in bioinformatics and the second for any data in tabular form. These two tools have in common that they are multimodal; conversation is supported by graphical content to guide users in complex tasks such as data science operations. In fact, we juxtaposed the conversation with a graphical user interface. The two interfaces are synchronous, and the behavior on one implies a state update of the other as well.

These two experiences led us to reflect on the effectiveness and potential of multimodal conversational interfaces, realizing, however, two gaps in the literature (Chapter 4): the lack of conceptualization of the integration of the various modalities and, consequently, the lack of a reference model for the design, specification, and implementation of such interfaces.

My work then proceeded by defining the "Multimodality Continuum" (Chapter 4), i.e., the spectrum of possible degrees of modality integration and with an analysis of the implications that multimodality brings within the interaction, both from linguistic (Chapter 6) and an interaction design perspective (Chapter 5): on the one hand, how the linguistic register changes as graphical elements are juxtaposed with the conversational agent, and on the other hand, how the designer must take the multimodal setting into account within experience design.

I then moved on to the formulation of an actual model for describing process-based multimodal conversational agents, explained in Chapter 7 defining the elements necessary for interface description and proposing a high-level architecture for the implementation of such interfaces. Finally, after creating and testing a multimodal interface in the educational domain in Chapter 8, I created an authoring tool that allows rapid programming and deployment of backend structures for managing such interfaces (Chapter 9).

1.2 Contributions

My work advances the state of the art in the design of multimodal conversational interfaces, providing design guidelines, empirical evidence, models, tools, and instruments that approach those interfaces at 360 degrees.

Every chapter is a separate study that presents its contributions, answering well-defined research questions. Overall, those contributions can be classified as follows:

- A detailed *Literature Analysis*, that provides the theoretical grounding for each work. In addition, a detailed meta-analysis of the findings is the basis for the elicitation of the theoretical models formulated in Chapter 2, design principles presented in Chapter 5, and the design of the authoring tool in Chapter 9;
- A set of *Theoretical Models*, in the form of the ontology presented in Chapter 2, the "Multimodality Continuum" introduced in Chapter 4, and the conceptual model for designing strongly-integrated multimodal conversational interfaces (Chapter 7);

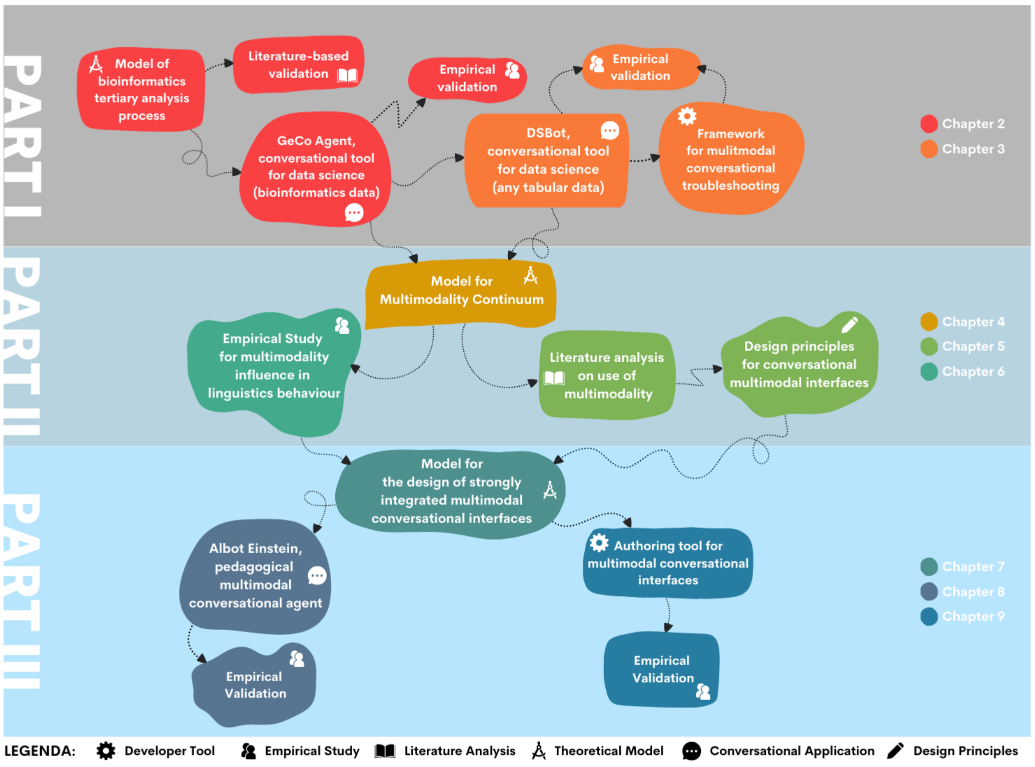

PART I
PART II
PART III
Model of bioinformatics tertiary analysis process
Literature-based validation
Empirical validation
Empirical validation
GeCo Agent, conversational tool for data science (bioinformatics data)
DSBot, conversational tool for data science (any tabular data)
Framework for mulitmodal conversational troubleshooting
Chapter 2
Chapter 3
Model for Multimodality Continuum
Empirical Study for multimodality influence in linguistics behaviour
Literature analysis on use of multimodality
Design principles for conversational multimodal interfaces
Chapter 4
Chapter 5
Chapter 6
Model for the design of strongly integrated multimodal conversational interfaces
Albot Einstein, pedagogical multimodal conversational agent
Authoring tool for multimodal conversational interfaces
Empirical Validation
Empirical Validation
Chapter 7
Chapter 8
Chapter 9
LEGENDA:
Developer Tool
Empirical Study
Literature Analysis
Theoretical Model
Conversational Application
Design Principles

FIGURE 1.1 – *Graphical representation of the work presented in the thesis*

- Three *Conversational Applications*; two of them for the analysis of supporting people in data science operations, and a pedagogical conversational agent to teach children chemistry, precisely the pH concept;
- Five *Empirical Studies* with real users to test the applications (Chapter 2, 3, 8, and 9) and to understand how the introduction of multimodality affects the interaction (Chapter 6);
- Two *Tools for Developers*, specifically an open-source framework to support multimodal conversational troubleshooting, presented in Chapter 3, and a graphical authoring tool to define and deploy backend structures of multimodal conversational applications, shown in Chapter 9.
- Some *Design Principles*, extracted from the analysis of the literature in Chapter 5, to be integrated with the contributions resulting from analysis of the qualitative interviews performed in the presented studies (Chapter 2, 3, 8, and 9).

Figure 1.1 graphically maps the types of contributions to the chapters.

1.3 Thesis Organization

This thesis is divided into three main parts, corresponding to my work's three main phases. Each chapter is developed as a research paper, and contains all the elements that make it usable also individually. Figure 1.1 graphically represents the flow of the various works presented in this Thesis.

In **Part I**, I will explore the use of multimodal conversational agents for data science, presenting two tools with this aim.

In **Chapter 2**, I will talk about GeCoAgent, a conversational interface for Bioinformatics Tertiary Analysis. Chatting with GeCoAgent users can filter data from heterogeneous sources and combine their pipelines only using natural language. Multimodality provides feedback and support coherently with the conversation. The design of the tool has been based on an ontology-based representation of the Bioinformatics Tertiary Analysis process, first elicited from user interviews in the form of a Hierarchical Task Tree and then formulated in the ontology formalism and validated through a literature analysis. The design and implementation of the conversational engine are based on the definition of grammar-based rules. We assessed the usability of the platform through a user testing session with 14 participants. The work presented in this chapter has been published in:

[11] Pietro Crovari, Sara Pidò, Pietro Pinoli, Anna Bernasconi, Arif Canakoglu, Franca Garzotto, and Stefano Ceri. 2021. Gecoagent: a conversational agent for empowering genomic data extraction and analysis. *ACM Trans. Comput. Healthcare*, 3, 1, Article 3, (October 2021), 29 pages. ISSN: 2691-1957. DOI: `10.1145/3464383`. `https://doi.org/10.1145/3464383`

[12] Sara Pidò, Pietro Crovari, and Franca Garzotto. 2021. Modelling the bioinformatics tertiary analysis research process. *BMC bioinformatics*, 22, 13, 1–27

[13] Pietro Crovari, Sara Pidò, and Franca Garzotto. 2020. Towards an ontology for tertiary bioinformatics research process. In *International Conference on Conceptual Modeling*. Springer, 82–91

[14] Pietro Crovari, Fabio Catania, Pietro Pinoli, Philipp Roytburg, Asier Salzar, Franca Garzotto, and Stefano Ceri. 2020. Ok, dna! a conversational interface to explore genomic data. In *Proceedings of the 2nd Conference on Conversational User Interfaces*, 1–3

In **Chapter 3**, I will present DSBot, another conversational agent for data science analysis. As I will discuss in the chapter, DSBot can be considered the evolution of GeCoAgent for many reasons, primarily for the extension to all data in tabular form and a different use of the conversation that is no more the central locus of the interaction but complements with the graphical interface. DSBot is composed of two main functional parts that correspond to the phases of the interaction: i) an engine that receives as input a dataset and a research question in natural language and automatically creates a data science pipeline that exploits AutoML techniques to produce an answer to the research question, and ii) a multimodal conversational troubleshooting module that supports users in the refinements of the results. In addition, we distributed the second module as a lightweight open-source module to be integrated into any web application. We empirically tested the efficacy of the analysis module, comparing our performances with the ones of well-established AutoML libraries and assessing the goodness of the translation capabilities of the system from the research question to the operative pipeline. We also tested the usability of the troubleshooting support through an empirical evaluation with 12 participants. Part of the work presented in this chapter is published in:

[15] Sara Pidó, Pietro Pinoli, Pietro Crovari, Francesca Ieva, Franca Garzotto, and Stefano Ceri. 2023. Ask your data—supporting data science processes by combining automl and conversational interfaces. *IEEE Access*, 11, 45972–45988. DOI: `10.1109/ACCESS.2023.3272503`

[16] Giulio Antonio Abbo, Pietro Crovari, Sara Pidò, Pietro Pinoli, and Franca Garzotto. 2022. Mctk: a multi-modal conversational troubleshooting kit for supporting users in web applications. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*, 1–3

[17] Giulio Antonio Abbo, Pietro Crovari, and Franca Garzotto. 2022. Enhancing conversational troubleshooting with multi-modality: design and implementation. In *International Workshop on Chatbot Research and Design*. Springer, to appear

In **Part II**, I will analyze the role of multimodality in the interaction from a problem framing, linguistic, and design perspective.

Chapter 4 will open with some reflections on the role of multimodality in the success of GeCoAgent and DSBot. Then, I will use that discussion as a starting point to frame the problem of multimodality in conversational interfaces. I will introduce the design dimensions and the problem space on which the rest of the thesis will discuss. I will introduce the "Multimodality Continuum", a taxonomy to describe the possible degrees of integration of conversational agents with other modalities. Finally, I will discuss the research questions that lead the work in the following chapters.

Chapter 5 will analyze the design implications of the introduction of multimodal elements in the interaction. The work followed a literature-based approach: we conducted a survey of the research works in the fields of multimodality with a particular focus on multimodal conversational applications, and we clustered the recurrent topics, eliciting seven design principles for such interfaces. The chapter ends with a case study in which we show how the design principles are applied to the design of GeCoAgent. The design principles have been published in:

[18] Pietro Crovari, Sara Pidó, Franca Garzotto, and Stefano Ceri. 2020. Show, don't tell. reflections on the design of multi-modal conversational interfaces. In *International Workshop on Chatbot Research and Design*. Asbjørn Følstad, Theo Araujo, Symeon Papadopoulos, Effie L.-C. Law, Ewa Luger, Morten Goodwin, and Petter Bae Brandtzaeg, editors. Springer. Springer International Publishing, Cham, (November 2020), 64–77. ISBN: 978-3-030-68288-0. DOI: `10.1007/978-3-030-68288-0\_5`

Chapter 6 will assess multimodality influence from the linguistic perspective in a comparative study. We set up an empirical study with 185 participants, who had to interact with multiple conversational agents, equivalent in the conversations, but different in the interaction design: some of them were uni-modal chatbots, others embedded graphical hints (such as buttons, carousels, and sliders) in the interface. We compared the performance of the users in the different conditions, observing the language use, the interaction time, and the number of errors committed.

In **Part III**, we will concentrate on modeling and implementing multimodal conversational agents, building on the knowledge generated in the previous part. I will start formulating a model to describe the design of those interfaces and then provide a complete example and realize an authoring tool to convert model-based specifications into the executable backend of those applications.

In **Chapter 7** I will describe the model we formulated for describing multimodal, process-based, task-oriented conversational interfaces. Our model takes inspiration from BPMN notation [19] to decompose the interaction process in tasks connected by gateways that rule the order of execution. We built this model pursuing the separation of concerns, that is, dividing the description of *what* users must do from *how* they can do it so that modifications of the UX (e.g., a change in the dialogue) does not affect the other modalities or the task specification.

Chapter 8 will present a detailed description of the case study I used to describe how the model works. I will introduce Albot Einstein, a multimodal pedagogical conversational agent thought to teach children chemistry, precisely the concept of pH. I will describe the design process that leads to the construction of the application, involving teachers and communication experts to elicit the design requirements. We conducted an empirical evaluation with 28 children from a secondary school, divided into two experimental conditions, to assess the learning effectiveness of such an application with respect to "traditional" web applications, showing comparable results.

Finally, in **Chapter 9**, I will present the authoring tool we developed to support developers in the creation of multimodal conversational interfaces. The tool we propose presents a drag-and-drop interface on which developers can design and define the behavior of the application they want to implement, with a formalism that is derived from the one described in Chapter 7. We also decided to include the management of the conversation inside the tool; developers can provide examples to train the Natural Language Understanding unit from a dedicated pane. Then, with the simple push of a button, the tool checks the correctness of the schema and deploys it into a working backend, ready to be integrated with a frontend. We tested our system with 15 developers that considered it a valid instrument.

The document ends with **Chapter 10** with a broader discussion on the contributions and the conclusions.

Part I

Multimodal conversational agents for data science

Chapter 2
GeCoAgent

2.1 Introduction and Research Questions

My Ph.D. started with a simple but compelling research question: propose a tool to support biologists and clinicians in genomic analysis.

Genomics is the discipline that studies the genome of an organism, that is, the genetic information that controls all the biological processes inside a living being, regulating its life, its growth, and its development. Genomics' primary focus is to investigate how the mutations of portions of DNA, i.e., modifications during its reproduction, are related to disease development.

Despite this discipline was born in the second half of the XX Century (the term genomics was used for the first time in 1987 by McKusick et al. [20], genomics saw exponential growth in the last two decades [21], when the commercial availability of Next Generation Sequencing techniques allowed researchers to sequence DNA, i.e., "read" its sequence of base pairs, at an accessible cost [22].

This increasing popularity leads experiments to produce enormous amounts of data at an unprecedented rate [23]. Clinicians, through tertiary analysis, can pursue precision medicine using this data as a supplement to existing clinical knowledge, while biologists can support genome research with a focus on tumors. This analysis is called *tertiary* because it is the last step of the bioinformatics pipeline, preceded by the formation of the sequence read (i.e., primary analysis) and the alignment of those sequences on a reference sample (i.e., secondary analysis) [24, 25]. A schematic representation of the process is shown in Figure 2.1 To be employed for these purposes, genomic data must be interpreted and transformed into actionable knowledge. Bioinformatics supported this requirement with several instruments: a massive amount of computational methods have been developed to support biologists and clinicians in their purpose. These tools are various in the technology they adopt and the problem they address, from the cloud and high-performance solutions to web-based orchestration platforms.

Genomic data is not only enormous but also complex [26]. Data originate from several processes, come from different sources, encode heterogeneous signals, and are presented in many different representations. For this reason, the ERC Advanced Project GeCo, led by Prof. Stefano Ceri, designed and implemented tools to integrate those data and simplify information extraction [25, 27–29].

However, bioinformatics tertiary analysis is still extremely challenging. To be able to draw meaningful conclusions, a researcher must formulate a valid research question and translate them into proper search routines, identify the right tool to extract the information

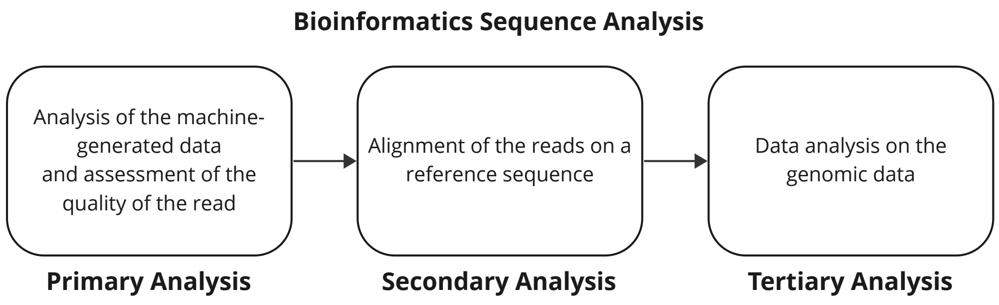

Bioinformatics Sequence Analysis
Analysis of the machine-generated data and assessment of the quality of the read
Alignment of the reads on a reference sequence
Data analysis on the genomic data
Primary Analysis
Secondary Analysis
Tertiary Analysis

Figure 2.1 – *The three steps of bioinformatics analysis*

needed to answer those questions, select a computational algorithm that can lead to the formulation of the thesis, understanding its functioning and its constraints; operate on data to make them suitable to be used with the selected algorithms; understand the numerical solutions proposed by the algorithm and, finally, interpret them in the biological domain to verify or discard the research hypothesis [30]. Biologists usually need to gain profound computer science knowledge; therefore, bioinformatics tools are often inaccessible.

The usability of bioinformatics tools is a serious issue that hinders both their capacity to support bioinformatics research and their potential for adoption, as Bolchini noted in [31]. Although many of these tools are highly difficult to learn and use, they were built to help bioinformatics operations and reduce the inherent challenges in the field. Managing them demands a large amount of cognitive work if the user is not an expert in both biology and computer science, which should be used to address research concerns. When biological operations and data become more complicated, this issue becomes worse. They call for sophisticated algorithms and computer science techniques that are frequently only known to machine learning software or data mining specialists.

With GeCoAgent, we try to overcome this limitation designing a big data service tailored to biologists' needs, particularly for professionals with limited computer science expertise that usually require computer scientists' support to perform data analysis with existing bioinformatics tools.

We decided to build GeCoAgent as a conversational interface, the first of its genre in the panorama of bioinformatics tool for tertiary research, to increase the accessibility of such a platform. In this way, researchers can dialogue with the platform instead of being required to write pieces of code. GeCoAgent transforms the conversation into a pipeline of data science operations and returns the computational results to users. To further increase the accessibility, we embed the conversational agent into a multimodal dashboard, where the user displays several visualizations concerning the operations performed interactively.

Before designing the interaction, we must elicit the research process that GeCoAgent must support. We use task analysis to create an ontology of bioinformatics tertiary analysis research that describes all the significant operations that the process comprises and their precedence constraints. Then, we use this ontology as the base to create a conversation that abstracts both the main phases of a typical analysis:

1. *Data Extraction*, guiding the user to select the genomic data on which to perform the operation and filtering them as desired;
2. *Data Analysis*, guiding the user to apply statistical and machine learning algorithms to the selected datasets.

Multimodality ensures that, at every step of the process, the user is supported by visualization of tables and typical statistics that describe the dataset and the operations performed.

In this first experiment of a multimodal conversational interface, after having shown a technique to elicit the process to inform the design of a conversational UI, we show that such an interface can empower bioinformaticians with no coding experience to operate alone on data and reduces the time required to complete the task and the number of error with respect to the use of existing tools, both GUI and code-based, such as GenoSurf [28], Jupiter Notebooks [32] and R-Studio [33].

The work presented in this chapter has been published in [11–13].

2.2 Related Work

2.2.1 Tools for Bioinformatics

Over the years, many tools have been developed for bioinformatics tertiary analysis. The first ones were executable programs or simple scripts to be executed through the command line. Tools with a Graphical User Interface (GUI) started to be created as soon as research interest in the subject increased. GUI-based tools can be split into two primary groups: those that facilitate the development of research pipelines and those that accomplish a particular operation.

Tools for Research Pipeline

This category comprises all the tools that support users to formulate research pipelines, providing a graphical user interface to integrate different modules into a unique workflow.

OrangeBioLab [34] is an example of a visual interface for the visualization and analysis of data. Users can upload data and define their pipeline through a block interface. OrangeBioLab provides modules for machine learning, data mining, predictive modeling, and data visualization.

Instead, UCSC Xena [35] is a tool for comparing and visualizing data on various dimensions. Users can rapidly compare the differences in the observed dimensions thanks to its layout that aligns the data in columns.

Galaxy [36] framework is a powerful graphical environment mainly designed for secondary analysis, and provides the foundations for Globus Genomics [37], a tool to create research workflows graphically to manage and analyze genomic data. Since it is a web application, researchers can collaborate sharing their projects.

A modular system called GenePattern [38] offers a plethora of genomic analysis capabilities via a graphical user interface. Its modules can be accessed via a block-based environment in a browser or run via code, using Python Notebooks, or through the command line. The system's components are highly adaptable to tailor the tools to the particular challenge.

Tools for Specific Operations

These tools focus on precise operations and therefore are conceived to be used in sequence with other tools to complete the operations pipeline. The most relevant are BEDTools, Integrated Genome Browser, and Bioconductor.

BEDTools [39] is an interactive application to compute genome arithmetic, that is, the application of set theory on the genome.

Integrated Genome Browser [40] allows biologists to explore interesting patterns from the biological perspective in genomic datasets through visualizations.

Bioconductor [41] expands R - a programming language for statistics - to allow researchers to analyze and comprehend high-throughput genomic data.

2.2.2 Conversational Agents for Data Science

Conversational technology can ease the use of data science tools, guiding users in the interaction with the system and abstracting the complexity of the underlying interface or programming language.

Many attempts have been made in Data Visualization. [42–44] are some examples. Tory thoroughly examines the effects of multimodality, emphasizing the significance of a strong interaction between the visualizations and the chat. The user studies reinforced the theory affirmed by [45] on the value of mixed interfaces by highlighting the significance of generating the visuals in conventional interfaces, thus segregated from the discussion.

The Data Retrieval process might also benefit from the use of conversational technology. Microsoft English, Precise, and Athena are among the most famous dialogue interfaces that transform a conversation into SQL-like languages [46–48]. BiographBot [49] is a dialogic interface for bioinformatics data retrieval. It operates through ALICE framework [50] to convert the conversation with the user into a query for the BioGraphDB [51], a genomic database based on Gremlin [52] query language that is freely accessible online. Maggie [53] is a conversational interface with a service-oriented architecture that is intended to obtain various biological data sets from BioCatalog [54] However, with Maggie, the user needs to be helped by the dialogue flow.

Finally, Iris [55] and Ava [56] try to use conversation for Data Analysis. Both of them focus on the construction of data analytics pipelines. Ava operates on a defined structure of the pipeline. Every step is a set of straightforward or nested operations that users must complete by inserting the parameter they need through the conversation. Ava then transforms the conversation into an executable Jupiter Python Notebook that guarantees the experiment's reproducibility. The conversation that acts as a Controlled Natural Language Interface [57] is based on a finite state machine. The verbal engagement with Ava is demonstrably more effective than the conventional notebook interface, according to experimental validation of the platform. On the other hand, the Ava system does not provide data exploration since the underlying state machine only offers linear interactions.

Iris Conversational Agent follows Ava's lead and converts user dialogue into callable Python routines. The authors develop a system where discrete functions may be stacked and integrated through a conversation, drawing on the premise that human interactions are built on a combination of atomic speech acts [58]. The Iris conversational engine creates a syntax tree that describes the chosen operations as the conversation progresses, and this syntax tree is ultimately translated into executable code. Additionally, Iris' empirical analysis demonstrated the effectiveness of such an interface by demonstrating how much faster the user could complete the given tasks. Since the system's atomic speech actions work as wrappers for the existing Python functions, the Iris functional structure ensures a far better level of user expression flexibility at the expense of more user programming proficiency.

2.2.3 Elicitation and Modeling of Tasks Requirements

With an in-depth knowledge of what users must do to accomplish their tasks, it is possible to design good and purposeful applications [59]. Consequently, a clear definition of all the users' tasks is necessary before designing a new interactive system. The Human-Computer Interaction community calls this process Task Model Elicitation[60].

Researchers developed many methodologies to fulfill this goal. GOMS [61] is one of the most adopted ones. The name is the acronym of the four phases that constitutes any task: Goals, Operators, Methods, and Section Rules. GOMS predicates that every user task can be described in these four elements. In addition, tasks might be hierarchically defined, iteratively describing the high-level ones as the composition of multiple sub-tasks until all the sub-tasks are considered atomic elements of the process. The intrinsic result of this decomposition process is a hierarchical tree of tasks, in which children nodes represents the sub-tasks that compose the parent tasks. GOMS framework has been declined in many variants, among which we find TADEUS [62], MECANO [63], TRIDENT [64], and MOBI-B [65]. However, the knowledge users acquire and use in the process is not modeled in these formulations as a task component.

ConcurTaskTree [66] is another popular methodology that describes the temporal relations, together with the structural relationships among the tasks.

Ontologies are another widely-used instrument in computer science. Ontologies are a formal description of concepts and how they relate to each othe, considered a valid instrument to describe phenomena [67]. Ontologies allow scientists to create a common ground with peers on which they create domain knowledge and share information [68]. They are particularly useful in complex domains such as biology and bioinformatics [69]. In particular, in this last community, the adoption of ontologies started with the formalization of the biomolecular field and since then has been widely recognized as valuable [70].

OBI, The Ontology for Biomedical Investigations, is one of the biggest ontologies in bioinformatics, counting more than 3600 classes and 100 properties [71]. As the name states, OBI describes the whole biomedical research process, from acquiring the data to processing and producing corresponding genomic data. The adoption of OBI allows researchers to share a common terminology when dealing with data and facilitate interoperability between data sources. OBI is based on OWL2 [71] and inherits the class types from the Basic Formal Ontology (BFO) [72], and from the Information Artifact one (IAO) [73]: OBI classes can be processes, material entities, processes and roles, and information entities.

To the best of our knowledge, there have been no research works centered on eliciting the entire bioinformatics tertiary analysis process, even if many experts recognized the importance of funding the design of bioinformatics tools on models such as ontologies of the method [74, 75].

2.2.4 GeCo Data Repository

Before proceeding with the description of the ontology and GeCoAgent, it is necessary to frame the development of this tool in the GeCo project's panorama. As said, GeCo is an ERC dedicated to analyzing genomic data. Since 2015, GeCo's research group has worked mainly on two complementary assets: a repository that integrates all the genomic and epigenomic data from the main public sources and the GenoMetric Query Language, a language to retrieve those data and perform complex operations on them.

GeCo Repository

GeCo Data Repository is a repository in which heterogeneous genomic (and epigenomic) data are integrated and shown uniformly, ready to be queried using the GenoMetric Query Language. Data are uniformed on the Genomic Data Model (GDM) [76], which describes both the clinical (and biological) information of the sample and the outcome of the biological experiments involving the sample itself. Precisely, each entry represents an experiment outcome, whereas the entry's metadata are key-value pairs that describe clinical information of the patient [27, 77, 78]. GeCo data repository is graphically accessible through GenoSurf, a tool to search and filter data graphically through a web interface.

GenoMetric Query Language (GMQL)

GMQL, the GenoMEtric Query Language, is a language to query genomics datasets represented in GDM [29]. Intuitively, the language consists of an algebra over datasets represented as region data and metadata, producing result datasets that contain the metadata of the input datasets on which they are dependent. The classical and domain-specific algebraic operations are the language's fundamental parts. GeCoAgent uses high-level data abstractions not concerned with algorithmic or formatting concerns characteristic of bioinformatics languages to convey the semantics of computations. This language organization transfers extremely well to these high-level data abstractions.

2.3 An Ontology to Describe the Bioinformatics Tertiary Analysis Research Process

2.3.1 Motivations

Bioinformatics Tertiary Analysis is a broad subject that comprises all the (complex) computer science methods, tools, and algorithms to extract biological knowledge from sequencing data coming from raw genomic data [25, 79].

To the best of our knowledge, most bioinformatics applications were designed to adopt a "system-centric" approach, meaning that the demands of the user and the nature of the involved human activities were given less attention than the technology requirements. A more "user-centric" approach that considers the viewpoint of the bioinformatics researcher from the very beginning of the technology design process would be required to make these apps more usable and helpful.

We claim that to pursue this shift, the design of bioinformatics tools should focus more on the following:

1. *the characteristics of the pipeline that is going to be used*; which are its components and which are their purposes;
2. *the inputs available to the user*; which are the input the user must provide and which are their structure (CSV tables, numeric parameters, configurations, etc.);
3. *the preferred and/or required outputs*; which are the outputs the operations will return, how they are structured, and how they should be structured to facilitate the researcher's work;
4. *the existing relations among the process elements*; looking at the operations not as isolated but as part of a broader process, therefore considering which outputs become inputs of other operations and which ones are necessary to continue the execution of the pipeline.

Only with a clear idea of the whole process it is possible to design a tool aware of the points mentioned above. For this reason, we create a formal description of the bioinformatics tertiary analysis process. Our method uses process models that are gradually improved to depict and make obvious all the stages often carried out by a bioinformatician in a typical tertiary analysis activity.

We first conducted an exploratory user study with eight bioinformaticians to elicit our model. They identified the tasks involved in tertiary analysis, conceptualized the process, and represented it as a Hierarchical Task Tree using a well-known technique called *hierarchical task analysis* [80]. Using the findings of a literature review containing examples of bioinformatics Tertiary Analysis, we then improved and validated the process model. Finally, using OWL [81], a common formalism for ontologies, we converted the final Hierarchical Task Tree into an ontology-based representation to increase its expressiveness.

Three contributions emerge from the elicitation of the ontology:

1. A Hierarchical Task Tree [80] of a outlining the processes of a tertiary analysis backed by examples demonstrating its thoroughness was developed after consulting many subject matter experts.
2. A formal ontological description of the resulting process requirements; this ontology, which is given in a common syntax, serves as a model for the reference process in bioinformatics. Tertiary analysis can be utilized by bioinformaticians to compare and analyze current tools as well as to create new ones.
3. An approach to elicit and model bioinformatics processes that might apply to technology design in various situations, especially those involving cognitively complicated cognitive tasks that require exact definition beginning with the identification of implicit expertise of the key actors.

2.3.2 From user interviews to a Hierarchical Task Tree

We elicited the ontology in a three-step process, shown in Figure 2.2. First, we interviewed eight bioinformaticians to understand the phases of their research work and we created a Hierarchical Task Tree to describe the tertiary analysis process. Then, we transformed the model adopting ontology formalism to enrich the representation with information about the precedence of the operations and the output at each step. Finally, we run a preliminary evaluation describing existing research processes with our ontology to assess the completeness of our model.

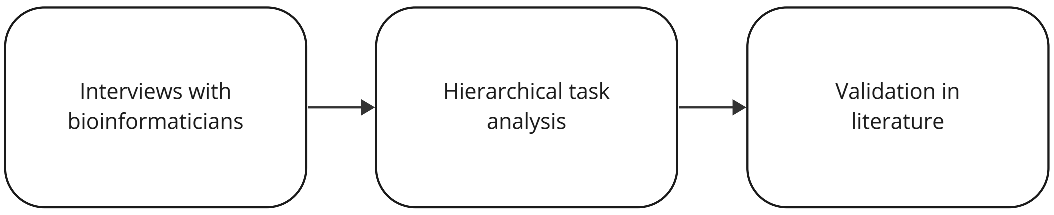


Figure 2.2 – *Schematic representation of the process that led to the formulation of the ontology*

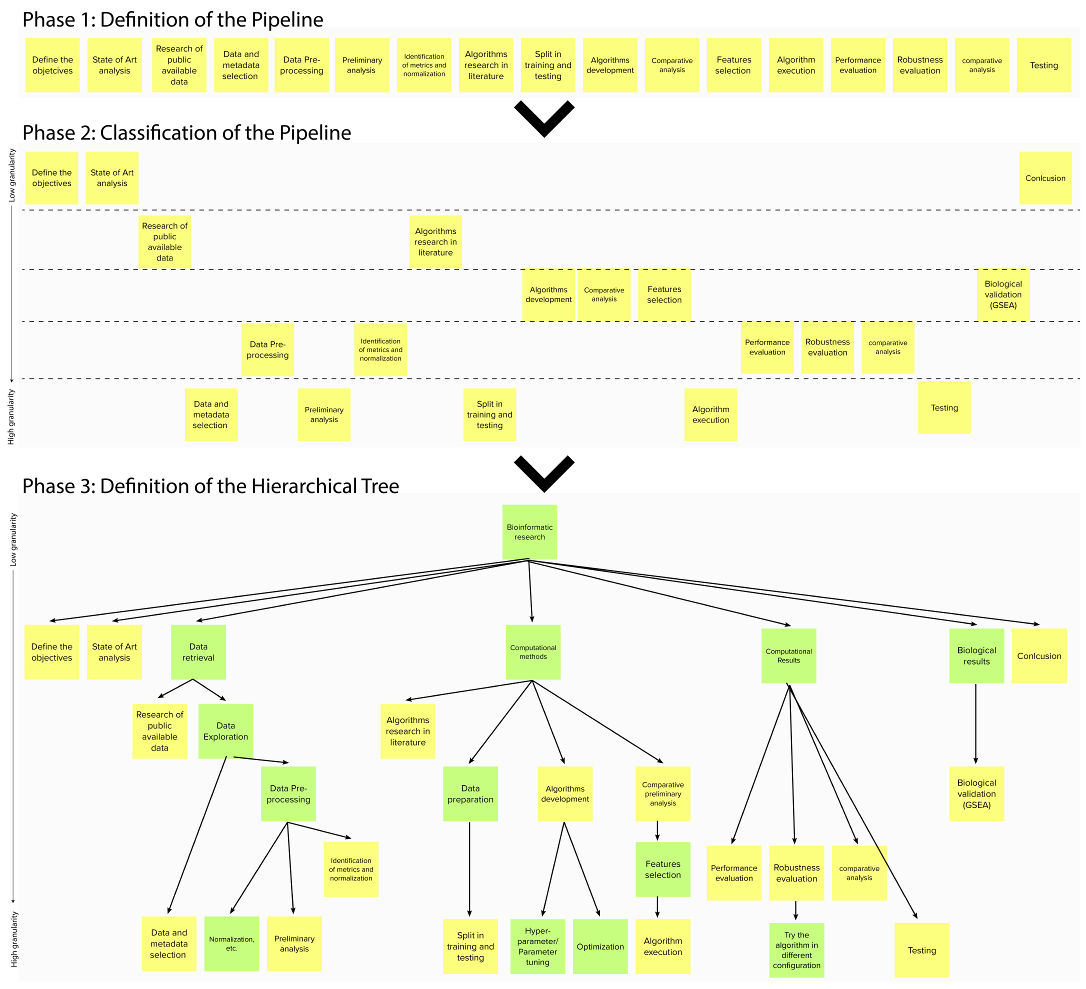


FIGURE 2.3 – *The three phases of the interview process. Source [12]*

Participants

We recruited eight bioinformaticians on a voluntary basis. Volunteers had different positions, reflecting their different levels of expertise: three Ph.D. students, two research assistants, two postdoctoral researchers, and an assistant professor. All of them considered bioinformatics tertiary research analysis as their primary field of research.

Setting

We run the interviews remotely. The interviewer and the bioinformatician were connected through a teleconferencing tool. The study participants had an online digital whiteboard with virtual sticky notes opened on their computers during the interview. The interviewer could access the same whiteboard on his machine. The whiteboard was different for every participant, so they could not be influenced by what had been said in the previous sessions.

Protocol

Each participant was involved in a interview session in that lasted around 45 minutes in which they were guided to elicit a Hierarchical Task Tree of their research process.

A Hierarchical Task Tree is a visual representation of tasks or subtasks organized in a hierarchical manner, where each task is broken down into smaller, more manageable parts. The tree structure starts with a high-level, overarching task, called the root node, which is then divided into multiple branches, each representing a sub-task. These sub-tasks can further be broken down into even smaller sub-tasks, creating a branching structure that allows for a clear and organized representation of a complex project or goal.

Because it gives a clear perspective of all the tasks involved, their interdependencies, and the sequence in which they must be accomplished, this structure is helpful for planning and managing projects. Additionally, it makes it simpler to assign resources and set deadlines and aids in identifying potential dangers or obstacles. As stakeholders can quickly comprehend the project's broad plan and their individual roles and responsibilities, it can also be used as a communication tool [82–84].

To aid the volunteer being questioned and to aid us with a visual viewpoint, we gave them access to a real or virtual whiteboard. We photographed the board after each stage to recreate the interview process when analyzing the findings. Each participant completed a consent form in which the study was fully described, along with a guarantee that the information gathered would be anonymous.

The interview session was divided into three phases, shown in Figure 2.3. The first phase consisted of the *Definition of the pipeline*. The participants were instructed to add a sticky note for each stage to the whiteboard as they described their usual research method in detail. No further restrictions were placed on the number of stages or the level of detail in the steps. During this initial phase, we allowed them as much leeway as possible and only interrupted them to ask for clarifications.

The interview continued with the *Classification of the pipeline*. Participants had to group their process components into several levels based on how abstract they were. They achieve this by layering their sticky notes on the board in accordance with the degree of abstraction. Since jobs were not subject to granularity restrictions during the previous phase, the pipeline was consistently highly diverse concerning the description's level of abstraction.

The third and final step, the *Definition of the Hierarchical-tree*, required the participants to construct a Hierarchical Task Tree of their usual research process by connecting the findings from the first and second parts of the interview. Participants could add sticky notes to complete the hierarchical structure if they believed it was necessary.

Results

All the participants successfully completed the interview process. We analyzed the outcome of the interviews by comparing the boards participants had composed with sticky notes at every step of the elicitation process, and integrating them with the notes taken during the interviews

The initial phases' results were comparable. Comparing the research flow and the higher degree of granularity reveals that, although the pipelines varied and each had its abstraction level, identical actions were still taken into account in the same sequence. We were able to instantly obtain four typical macro-phases common in every research process: *Data Retrieval*, *Data Exploration*, *Data Analysis*, and *Results Validation*. However, when we took a closer look at the outcomes of the interviews, we saw that every participant paid more attention to a different stage of the process. This gave us a detailed specification of each phase and allowed us to retrieve a complementary viewpoint on the tertiary bioinformatics study.

Assessing the results of the second part of the interview, i.e., the classification, we noticed that results varied because each participant had a different level of abstraction. Each participant's steps and subdivisions were unique. We noted that the people we spoke with assigned comparable abstraction levels to related activities. Almost all participants classified data using three or four distinct abstraction levels.

We then looked at the trees that were produced. Three major phases made up this analysis. The investigation of the topologies of the created trees came first. The interview set as a whole shared a similar fundamental structure. The topologies of the trees were varied, nevertheless, especially at the deepest nodes. This is because every participant's attention was on a different workflow stage. The nodes of the trees were then compared, and we made an effort to create a single tree with all of the complimentary and common nodes. There were hardly any conflicts found in this comparison. The tree's leaf nodes were the location of the few disagreements. This enables us to emphasize that the researchers implicitly concur with the typical methodology of a tertiary bioinformatics study. The next phase involved precisely comparing the generated tree to single ones. The remaining nodes in the interview trees were examined. Although few, we tried to incorporate them into the new framework and, if possible, integrate them.

2.3.3 A Hierarchical Task Tree to describe Tertiary Analysis

The presented approach produces a Hierarchical Task Tree that describes the tertiary research analysis process of bioinformatics.

We combined the trees produced by the participant interviews into a single structure to elicit the model. Resulting diagrams presented some inconsistencies. In those situations, we chose the course of action that the vast majority of the participants took. We requested an expert bioinformatician who had not been interviewed to settle the dispute and offer his point of view when an equal number of participants favored the opposing viewpoints.

This depiction works for a variety of reasons. First, the tree provides the process description at several levels of abstraction, offering the appropriate amount of granularity for the given issue. For this reason, systems that operate at many levels may all be described using the same paradigm. The *part of* relationship between parent nodes and children is embedded in the tree at the same time, giving the conditions essential for eliciting all of the actions that a tool must offer. In fact, a tool created for task *A* must support every action that task *A*'s child nodes define.

Even if formalisms such as ConcurTask Trees [66] are a more potent tool for task description, we chose to utilize Hierarchical Task Trees instead since they are the most similar representation of the user replies we gathered from the interviews. Additionally, this is simply a transitional representation before adopting ontology formalism, as detailed later in this article; it is not the final model.

Figure 2.4 displays the generated tree. All the participant stated that the tertiary analysis process could be broken down into the four major stages of *Objective Definition, Data Extraction, Data Analysis*, and *Results Analysis*, which are characteristic of most data science applications. Going deep into the structure, or examining the process at a more granular level, causes domain-specific distinguishing characteristics to emerge.

According to all study participants, Objective Definition—a description of what a researcher hopes to learn and get from that analysis—is where the bioinformatics research process begins. This task is divided into three parts: *Research Question Definition, State of the Art Analysis,* and *Deliverables Definition*, or the questions the researcher wants to answer, an analysis of the works that are relevant to that subject, and a characterization of the outcomes the researcher intends to present after the pipeline. The last stage is often carried out in collaboration with domain specialists who will assess the findings from a biological standpoint. A collection of tables, charts, or data required to substantiate the research hypothesis produced in the *Research Question Definition* step is the outcome of the deliverable definition.

Data Extraction starts when the research's goal has been established. *Data Retrieval, Data Exploration*, and *Data Integration* are the other three components of this phase. In order to determine which of the accessible data sets may be utilized to address the research issue, *Data Retrieval* starts with a *Search of Publicly Available Datasets*. The *Dataset Selection* brings this process to a conclusion. The scientist initially interacts with the chosen data during the *Data Exploration* phase. A *Preliminary Analysis* is initially performed to get a basic knowledge of the data. This analysis is carried out by conducting a *Literature Research on Data*, evaluating the data through a *Data Assessment*, and then performing a *Format Check*. Then, *Data Pre-processing* is carried out to reduce data noise. A preliminary round of Quality Assessment is completed to comprehend noisy data. The process of *Quality Correction and Data Cleaning* begins. It consists of three steps: *Bad Data Discard, Missing Data Imputation*, and *Trimming* to remove extreme values and/or outliers. *Data Normalization*, which is broken down into *Metric and Normalization Method Identification* and *Values Normalization*, marks the end of the pre-processing stage. *Data Visualization* is crucial to comprehending the nature of the data in the analysis when exploring the dataset. *Visualization Method Identification* and *Visualization Creation* are the two aspects of this process. *Data Integration* marks the end of the extraction process. Heterogeneous data are combined in this step to create a unique dataset on which to conduct the analysis. Integration is the end product of three consecutive processes: *Literature*

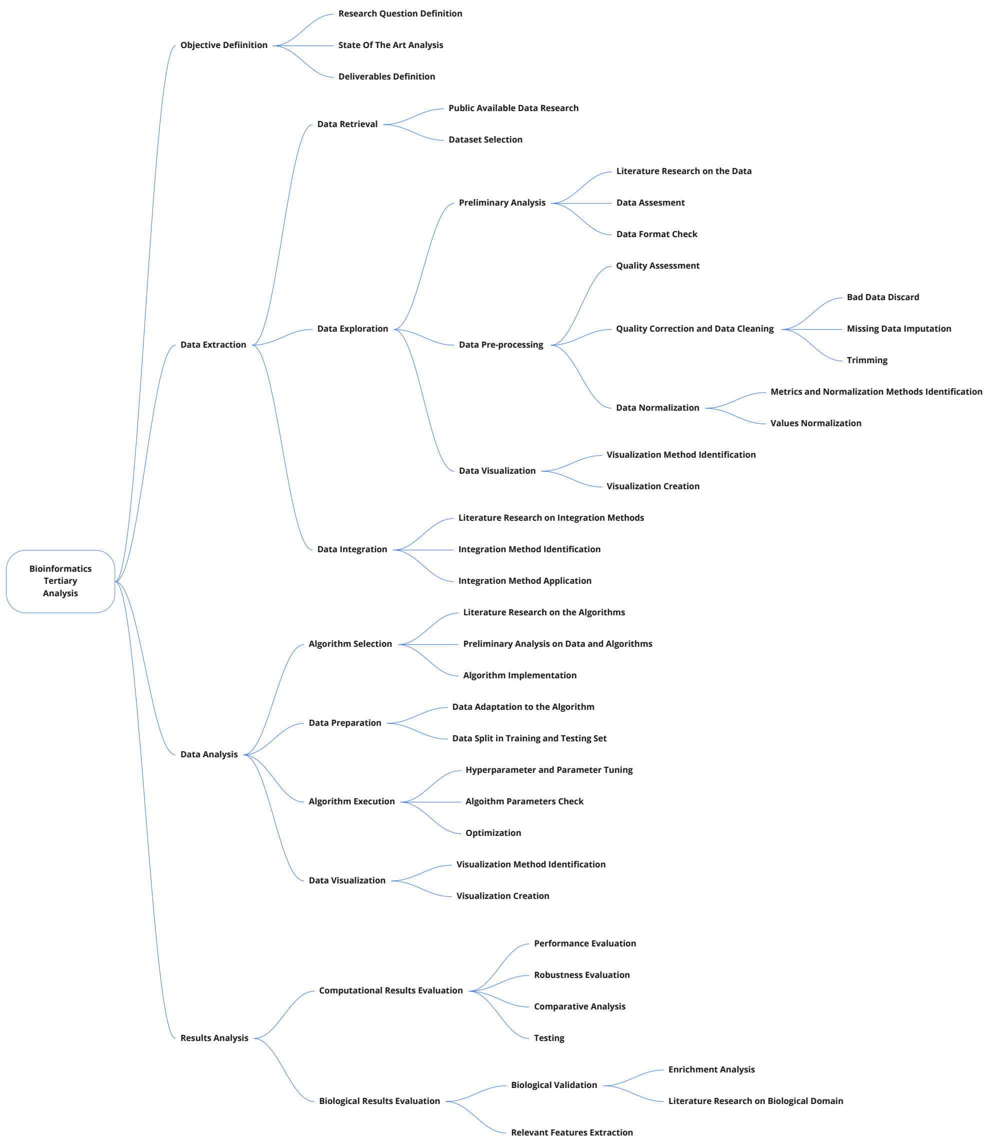

Bioinformatics Tertiary Analysis
Objective Defiinition
Research Question Definition
State Of The Art Analysis
Deliverables Definition
Data Extraction
Data Retrieval
Public Available Data Research
Dataset Selection
Data Exploration
Preliminary Analysis
Literature Research on the Data
Data Assesment
Data Format Check
Data Pre-processing
Quality Assessment
Quality Correction and Data Cleaning
Bad Data Discard
Missing Data Imputation
Trimming
Data Normalization
Metrics and Normalization Methods Identification
Values Normalization
Data Visualization
Visualization Method Identification
Visualization Creation
Data Integration
Literature Research on Integration Methods
Integration Method Identification
Integration Method Application
Data Analysis
Algorithm Selection
Literature Research on the Algorithms
Preliminary Analysis on Data and Algorithms
Algorithm Implementation
Data Preparation
Data Adaptation to the Algorithm
Data Split in Training and Testing Set
Algorithm Execution
Hyperparameter and Parameter Tuning
Algoithm Parameters Check
Optimization
Data Visualization
Visualization Method Identification
Visualization Creation
Results Analysis
Computational Results Evaluation
Performance Evaluation
Robustness Evaluation
Comparative Analysis
Testing
Biological Results Evaluation
Biological Validation
Enrichment Analysis
Literature Research on Biological Domain
Relevant Features Extraction

Figure 2.4 – *Hierarchical Task Tree of the Bioinformatics Tertiary Analysis process*

Research on Integration Methods, the *Integration Method Identification*, and the *Integration Method Application*.

Data Analysis then starts. In order to extract information from the data, statistical and computational procedures are used to examine the data. *Algorithm Selection, Data Preparation*, and *Algorithm Execution* make up the three processes that make up the analysis. The best algorithm is selected during *Algorithm Selection*. This procedure is supported by a *Literature Research on the Algorithm* to determine state of the art in comparable works, a *Preliminary Analysis on Data and Algorithms* to determine whether the dataset and the chosen method(s) are compatible, and lastly the *Algorithm Implementation*. The dataset must be transformed during the *Data Preparation* step to execute the chosen algorithm on it. The *Data Adaptation to the Algorithm* and *Data Split into Training and Testing Set* are the first steps in doing that. To accurately evaluate the trained algorithms, the last step is crucial. The *Algorithm Execution* comes last. According to the algorithm, these phases involve a wide range of operations. They may be divided into three phases: *Hyper Parameter and Parameter Tuning, Algorithm Parameters Check*, and *Optimization*.

Results Analysis is the fourth and last stage of the bioinformatics tertiary research process. Here, knowledge is created from the information that the algorithms retrieved. To achieve that, it is first important to conduct a *Computational Results Evaluation* to determine their significance and whether they may be regarded as valid. *Performance Evaluation, Robustness Evaluation, Comparative Analysis*, and *Testing* make up the computational validation. The data are then subjected to *Biological Results Evaluation* to see whether a biological explanation is possible. Biological Results Evaluation consists of three tasks: *Biological Validation*, which is divided into *Enrichment Analysis* and *Literature Research on Biological Domain*, *Relevant Features Extraction*, and finally, *Functional Genomic Analysis*.

2.3.4 Validation

We conduct a literature-based investigation for the Hierarchical Task Tree to be validated. In particular, we aim to evaluate its descriptive power and identify the study work characteristics that this model emphasizes.

We methodically chose 30 research and technique publications in bioinformatics tertiary analysis from two sources: the latest work published in the *BMC Bioinformatics* magazine and the works from the Genomic Computing Group[1]. We scanned through the articles published on the journal, ordered by the most recent publication date, choosing on the title and abstract of the documents. The research has been carried out on December 8th, 2020.

We chose these two sources for precise reasons. BMC Bioinformatics is one of the most important journals in bioinformatics tertiary analysis domains. Every year it publishes high-quality works on a variety of sub-domains, allowing us to observe a good variety in the works. In addition, all the articles are open access. We adopted the paper published by our

[1]http://www.bioinformatics.deib.polimi.it/geco

research group as second source, so that in case of doubts on the methodologies followed in the studies, we could directly contact the authors.

We paid particular attention to current methods or research papers addressing tertiary bioinformatics analysis. In addition to eliminating all software-related publications, we also eliminated all papers that did not use the findings of secondary analysis as their point of departure. The papers were then carefully read and mapped onto the tree's 36 leaves. To be considered in the document, a task must be expressly specified. In Figure 2.5, two instances of this procedure are depicted. Table 2.2 displays the outcomes.

ID	Paper Title	Reference
P01	Latent-space embedding of expression data identifies gene signatures from sputum samples of asthmatic patients	[85]
P02	Analysis of genomic and transcriptomic variations as prognostic signature for lung adenocarcinoma.	[86]
P03	Association rule mining to identify transcription factor interactions in genomic regions. Bioinformatics.	[87]
P04	Designing and evaluating deep learning models for cancer detection on gene expression data.	[88]
P05	Investigating deep learning based breast cancer subtyping using pan-cancer and multi-omic data.	[89]
P06	Matrix factorization-based technique for drug repurposing predictions.	[90]
P07	Latent Dirichlet allocation based on Gibbs sampling for gene function prediction	[91]
P08	Spatial patterns of CTCF sites define the anatomy of TADs and their boundaries	[92]
P09	NAUTICA: classifying transcription factor interactions by positional and protein-protein interaction information.	[93]
P10	Network modeling and analysis of normal and cancer gene expression data.	[94]
P11	Combining DNA methylation and RNA sequencing data of cancer for supervised knowledge extraction.	[95]
P12	Modeling gene transcriptional regulation by means of hyperplanes genetic clustering.	[96]
P13	Exploiting ladder networks for gene expression classification.	[97]
P14	CGINET: graph convolutional network-based model for identifying chemical-gene interaction in an integrated multi-relational graph.	[98]
P15	Deep learning based DNA: RNA triplex forming potential prediction. BMC Bioinform.	[99]
P16	Prediction and prioritization of autism-associated long non-coding RNAs using gene expression and sequence features.	[100]
P17	Prediction of enhancer-promoter interactions using the cross-cell type information and domain adversarial neural network.	[101]
P18	SAlign–a structure aware method for global PPI network alignment.	[102]

P19	Improved cytokine-receptor interaction prediction by exploiting the negative sample space	[103]
P20	LAMP: disease classification derived from layered assessment on modules and pathways in the human gene network.	[104]
P21	Feature selection algorithm based on dual correlation filters for cancer-associated somatic variants.	[105]
P22	Variable selection from a feature representing protein sequences: a case of classification on bacterial type IV secreted effectors.	[106]
P23	Predicting MiRNA-disease associations by multiple meta-paths fusion graph embedding model.	[107]
P24	Cancer prognosis prediction using somatic point mutation and copy number variation data: a comparison of gene-level and pathway-based models.	[108]
P25	MCCMF: collaborative matrix factorization based on matrix completion for predicting miRNA-disease associations.	[109]
P26	Integration of anatomy ontology data with protein–protein interaction networks improves the candidate gene prediction accuracy for anatomical entities.	[110]
P27	Impact of data preprocessing on cell-type clustering based on single-cell RNA-seq data.	[111]
P28	Correntropy induced loss based sparse robust graph regularized extreme learning machine for cancer classification.	[112]
P29	Leveraging TCGA gene expression data to build predictive models for cancer drug response.	[113]
P30	Robust edge-based biomarker discovery improves prediction of breast cancer metastasis.	[114]

Results

The Hierarchical Task Tree is complete. Figure 2.6 displays the aggregate values or the frequencies of tasks and articles that were present. Every operation outlined in the paper may be mapped to a job in the tree, demonstrating the model's completeness.

Bioinformatics papers typically describe half of the tasks of the Hierarchical Tesk Tree. The average number of tasks stated is about half of the 36 tasks (avg: 18.9, standard deviation: 3.0). The low standard deviation demonstrates how the description's depiction of the number of tasks is homogeneous.

Not all the tasks are described with the same frequency in bioinformatics papers. The same logic does not hold for the tasks, as seen by the uneven distribution of their references. The fact that 10 tasks —*Research Question Definition*, *State of the Art Analysis*, *Dataset Selection*, *Literature Research on the Data*, *Data Assessment*, and *Data Format*

		Paper																													
ID	**Tree Leaf (Task Name)**	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**	**P12**	**P13**	**P14**	**P15**	**P16**	**P17**	**P18**	**P19**	**P20**	**P21**	**P22**	**P23**	**P24**	**P25**	**P26**	**P27**	**P28**	**P29**	**P30**
1	**Research Question Definition**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
2	**State of the Art Analysis**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
3	**Deliverable Definition**																														
4	**Public Available Data Research**		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
5	**Dataset Selection**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
6	**Literature Research on the Data**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
7	**Data Assessment**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
8	**Data Format Check**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
9	**Quality Assessment**			X		X		X	X									X	X								X			X	
10	**Bad Data Discard**			X		X		X	X	X	X		X			X									X		X				X
11	**Missing Data Imputation**			X				X																							
12	**Trimming**					X		X	X	X	X				X	X						X					X	X			X
13	**Metrics and Normalization Methods Identification**	X	X			X			X		X		X						X									X		X	X
14	**Values Normalization**	X	X			X			X		X		X						X									X		X	X
15	**Visualization Method Identification**		X		X	X			X		X			X	X										X		X	X		X	X
16	**Visualization Creation**		X		X	X			X		X			X	X										X		X	X		X	X
17	**Literature Research on Integration Methods**		X				X				X	X												X			X			X	
18	**Integration Method Identification**		X				X				X	X							X					X			X			X	
19	**Integration Method Application**		X				X				X	X							X					X			X			X	
20	**Literature Research on the Algorithms**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
21	**Preliminary Analysis on Data and Algorithms**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
22	**Algorithm Implementation**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
23	**Data Adaptation to the Algorithm**	X	X	X	X	X	X	X	X	X	X				X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
24	**Data Split in Training and Testing Set**	X	X		X	X		X						X	X	X	X	X	X	X			X		X					X	X
25	**Hyperparameter and Parameter Tuning**	X		X	X	X		X					X	X		X	X			X										X	
26	**Algorithm Parameters Check**	X		X	X	X		X		X			X	X		X	X			X										X	
27	**Optimization**						X	X		X												X				X	X		X		X
28	**Visualization Method Identification**	X	X	X		X			X	X	X		X		X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X
29	**Visualization Creation**	X	X	X		X			X	X	X		X		X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X
30	**Performance Evaluation**		X		X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X
31	**Robustness Evaluation**			X					X																				X		X
32	**Comparative Analysis**	X		X	X		X	X			X	X	X							X									X	X	X
33	**Testing**				X									X	X																
34	**Enrichment Analysis**	X	X					X	X	X	X								X												X
35	**Literature Research on Biological Domain**	X	X	X		X	X	X	X		X								X							X	X				
36	**Relevant Feature Extraction**	X				X						X	X																		

Table 2.2 – *Mapping of the leaf nodes of the Hierarchical Task Tree on the paper examined during the validation process.*

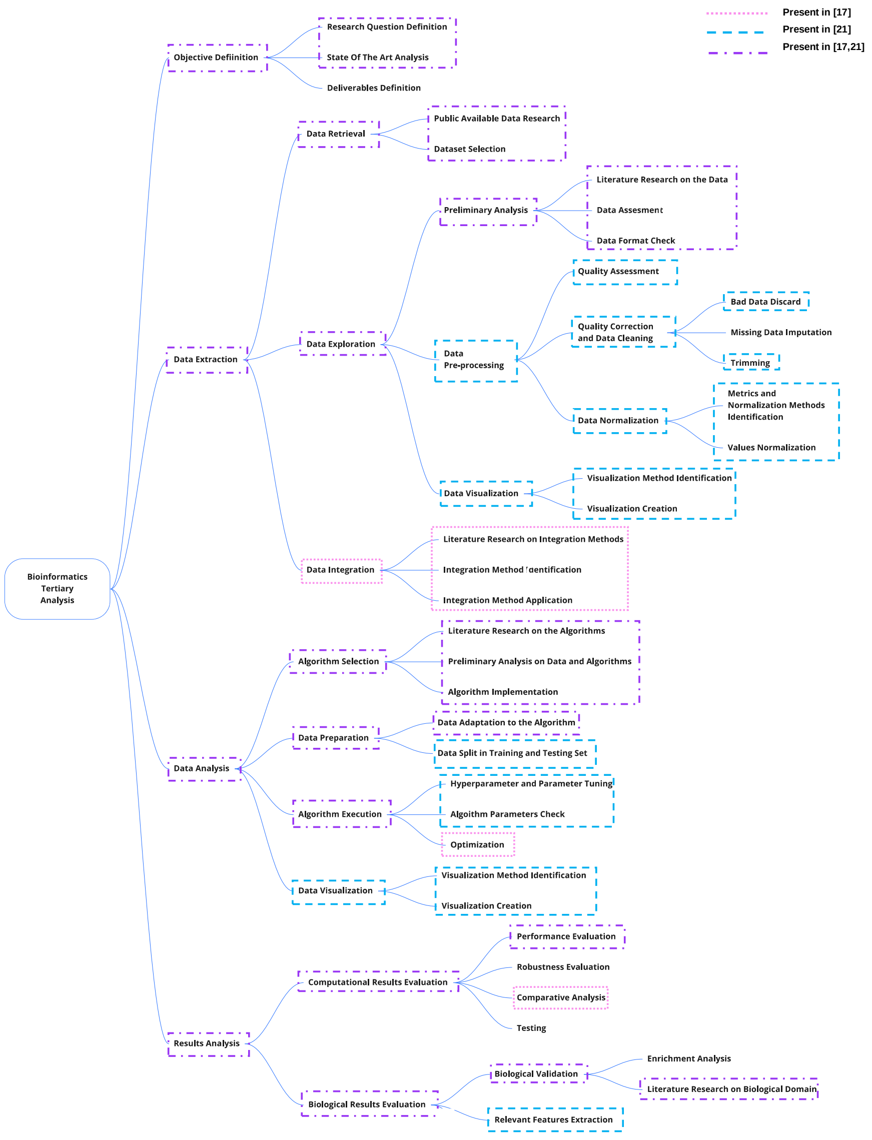

Present in [17]
Present in [21]
Present in [17,21]
Bioinformatics Tertiary Analysis
Objective Definition
Research Question Definition
State Of The Art Analysis
Deliverables Definition
Data Extraction
Data Retrieval
Public Available Data Research
Dataset Selection
Data Exploration
Preliminary Analysis
Literature Research on the Data
Data Assesment
Data Format Check
Data Pre-processing
Quality Assessment
Quality Correction and Data Cleaning
Bad Data Discard
Missing Data Imputation
Trimming
Data Normalization
Metrics and Normalization Methods Identification
Values Normalization
Data Visualization
Visualization Method Identification
Visualization Creation
Data Integration
Literature Research on Integration Methods
Integration Method Identification
Integration Method Application
Data Analysis
Algorithm Selection
Literature Research on the Algorithms
Preliminary Analysis on Data and Algorithms
Algorithm Implementation
Data Preparation
Data Adaptation to the Algorithm
Data Split in Training and Testing Set
Algorithm Execution
Hyperparameter and Parameter Tuning
Algoithm Parameters Check
Optimization
Data Visualization
Visualization Method Identification
Visualization Creation
Results Analysis
Computational Results Evaluation
Performance Evaluation
Robustness Evaluation
Comparative Analysis
Testing
Biological Results Evaluation
Biological Validation
Enrichment Analysis
Literature Research on Biological Domain
Relevant Features Extraction

Figure 2.5 – *Example of mapping the process described in two research papers on the Hierarchical Task Tree. Source [12]*

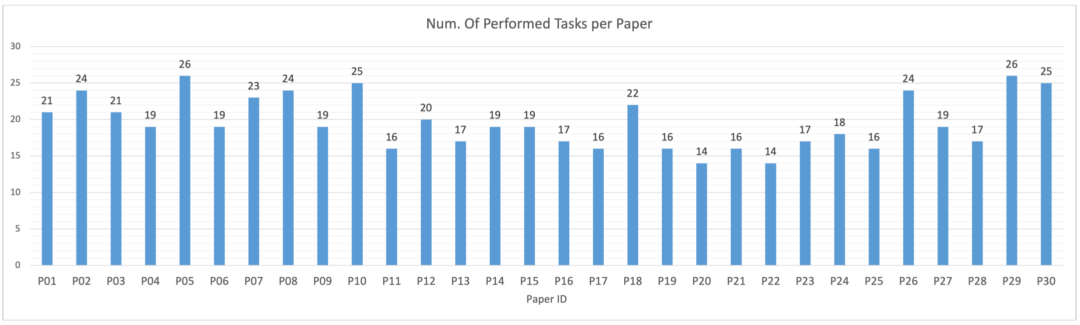


Figure 2.6 – *Number of tasks described in every paper examined in the validation process*

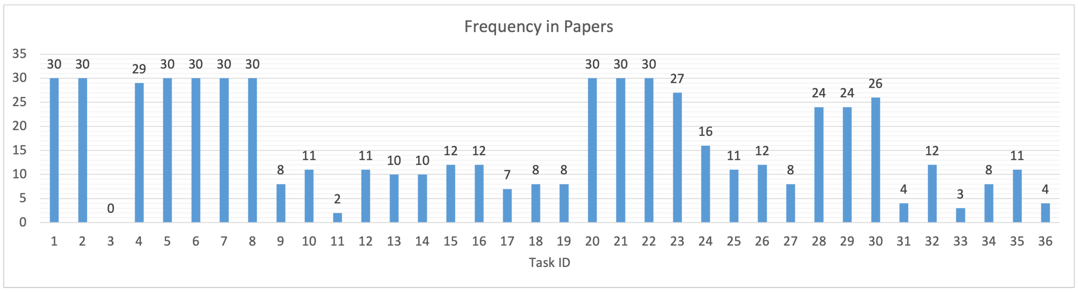


Figure 2.7 – *Number of occurrences of every task in the set of papers examined in the validation process*

Check, *Literature Research on the Algorithms*, *Preliminary Analysis on Data and Algorithms*, *Algorithm Implementation* and *Data Adaptation to the Algorithm*—are present in all the papers examined highlights their significance in the analysis. *Public Available Data Research* (28/30) and *Performance Evaluation* (26/30) are two of the jobs that get great attention as well. Also, these 2 tasks are important; the lack of this step in two documents is because one of those studies are based on datasets of previous works [85, 86], while the other leverage on other validation techniques like *Comparative Analysis* [87, 88] or *Robustness Evaluation* [85, 87].

Five activities are mentioned three times or less (*Missing Data Imputation, Robustness Evaluation, Testing*), indicating that not all tasks are as popular. The considered papers never specifically mention the *Deliverable Definition*. The reason is that although this phase is essential for a research project's success, it is frequently implied and not explicitly stated. Additionally, it was mostly ignored during the interview process (7/8). This procedure is essential when the results are confirmed by individuals who are not computer scientists, according to the lone researcher who made a note of it.

Sibling tasks tend to appear together in bioinformatics papers. Only 36 of the 503 instances of tasks appear isolated, that is, without the neighboring tasks being mentioned, when we look at how the tasks are presented. Additionally, the isolated tasks are not evenly distributed, although they are all part of the sub-tasks for *Quality correction and Data Cleaning* and *Results Analysis*. The reason is that quality repair and data analysis operations are different activities rather than phases in a single process. This tendency is much more pronounced in outcomes analysis; only a few studies utilize many forms of validation in the same tree branch.

Patterns emerge in tasks descriptions in bioinformatics papers. A more thorough study reveals that some neighboring rows (i.e., tasks) seem always to be coupled: *Visualization Method Identification* and *Visualization Creation*, *Metrics and Normalization Methods Identification* and *Values Normalization*, and the triple *Literature Research on Integration Methods*, *Integration Method Identification*, *Integration Method Application*. Similar to how knowing how to view data before plotting it validates this behavior, the semantics of the operations in this instance also support it. The first task of the couples is the prerequisite and essential task for the correct execution of the second one.

The presence of some tasks is a good indicator of the nature of the examined paper. Finally, a fascinating connection between Data Pre-processing procedures and Biological Results Evaluation is discovered. The nature of the research works justifies this link. Preprocessing the data is a common concern in papers that wish to infer biological inferences from the data. With this in mind, we can distinguish between two main types of contributions: *computational* ones, which concentrate research on a new algorithmic approach for tertiary analysis, and *biological* ones, which seek to discover both new computational techniques and new biological advancements from a biological perspective. With this in mind, we can distinguish between two main types of contributions: computational ones, which concentrate research on a new algorithmic approach for tertiary analysis, and biological ones, which seek to discover both new computational techniques and new biological advancements from a biological perspective. We discover that the number of tasks in the Data Pre-processing and Biological Results Evaluation is a good indicator of the category in which the manuscript falls: if the manuscript only has a computational scope, we can observe that the Data Pre-processing steps are almost always skipped, otherwise, at least one of them is almost always performed, and the Biological Results Evaluation is present.

2.3.5 Creating the Ontology

Despite the Hierarchical Task Tree being able to describe in detail all the tasks that compose the Bioinformatics Tertiary Process, this representation presented certain drawbacks. The relational component was insufficient to in-depth characterize the process. The tree-based description offers no details on the jobs' results. Furthermore, the hierarchical structure is too rigid to convey the order of duties adequately. In reality, for most siblings, the precedence order is rigidly determined by the tree's order of appearance. This is only true for some tree nodes: a depth-first visit technique is insufficient to identify job priority in some parts of the process when there is no definite precedence order, such as with Literature Research on the Data and its siblings. Last but not least, the Hierarchical Task Tree cannot be combined with other models or utilized with tools for their exploitation and analysis because it is not specified using the common declarative languages used for ontologies.

These factors led us to develop our model by adopting OWL2 Web Ontology Language's formalism [81]. We picked OWL over OBO [115] since the first offers greater semantic support than OBO. Our model is based on the Resource Description Framework (RDF) [116] representation of data, which specifies that subject-predicate-object triples should be used to code the model. As seen in the Figure, this representation implicitly generates a directed graph of the ontology. The RDF Schema [117] serves as the foundation upon which OWL bases its language, giving an expressive way to define the ontology components and their relationships. The outcome is a first-order logical piece that can be decided upon. In other words, we can create OWL reasoners that respond to model-related queries in a set amount of time and steps.

We used Web-protege [118], a web-based graphical user interface for modeling OWL-based ontologies, to translate the model. Figure 2.8 displays a visual depiction of the findings, while Figure 2.8 shows a section of it to have increased readability. The model's most recent version may be found in an online repository[2].

There are 3 relations and 70 classes in the resultant model. OBI classes *Information Content Entity* and *Planned Process* make up the higher level. Planned procedures are generally taken by a researcher and result in an information content entity as the outcome. Information Content Entities are the knowledge, information, and outputs produced by a process successfully carried out. Keep in mind that good execution does not guarantee a successful conclusion. No matter if the outcome confirms or contradicts the study hypothesis, we refer to procedures as *successful* when their technique is correctly carried out, and some result is achieved. The *Results* subclass, which is the parent of all the Information Entities that generate new information about the study topic, is one of the Information Content Entities.

The following three relations link classes together:

1. *Has Part* relation defines the sub-processes that compose a process. In the Hierarchical Task Tree, this relationship is comparable to the parent-child relationships. A sub-process shared by many processes can be the subject of numerous relations instead of the tree representation, eliminating the tree's duplication (e.g., Data Visualization). Any process may have zero or more *Has Part* relations.
2. *Has Specified Output* describes the output for a task. Due to this, in a relationship like this, the subject must be a planned process, and the object must be an information content entity. An Information Content Entity may target one or more Has Specified Output Planned Processes and may be the subject of zero or more Has Specified Output Planned Processes. The following property holds: given *A* and *B* Planned Processes objects and *C* being an Information Content Entity, if *(A has part B)* and *(B has specified output C)*, then *(A has specified output C)*. These inferred properties are not explicitly reported in the ontology.

[2]https://github.com/peempi/btap

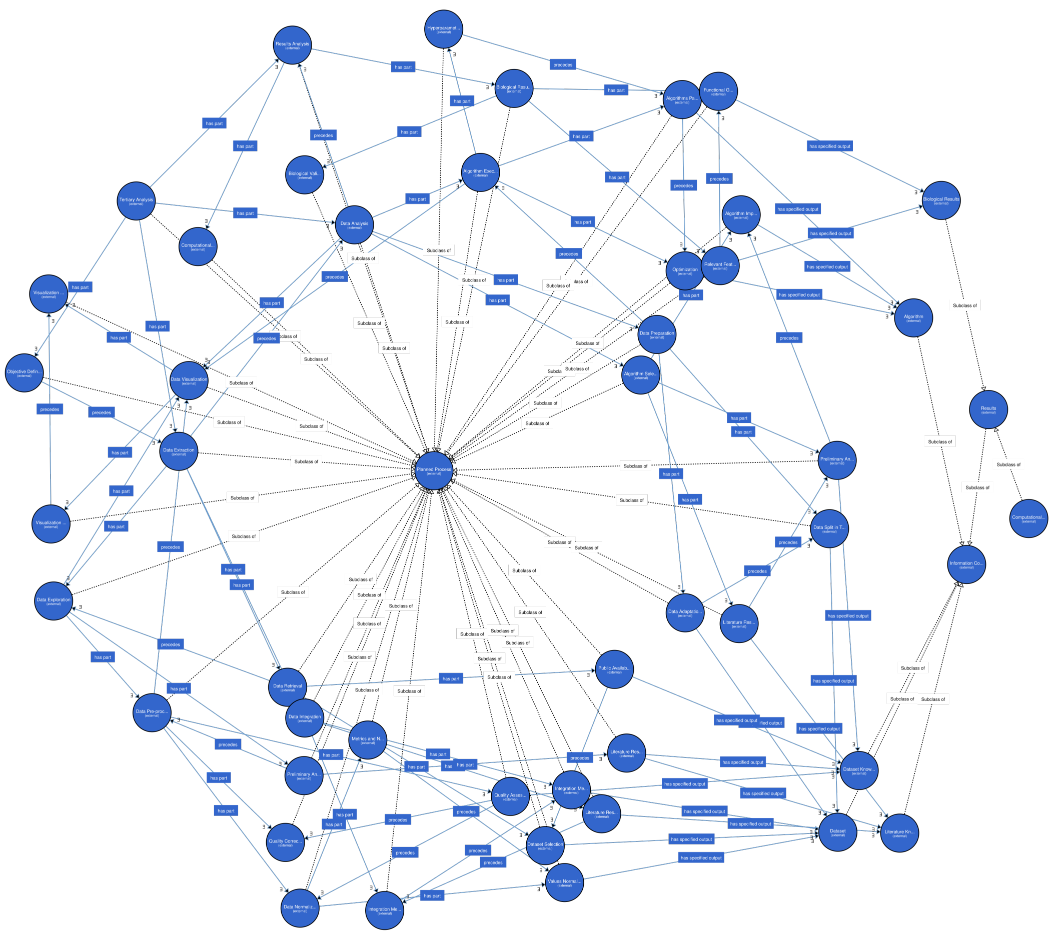


FIGURE 2.8 – *Graphical representation of the ontology. Source [12]*

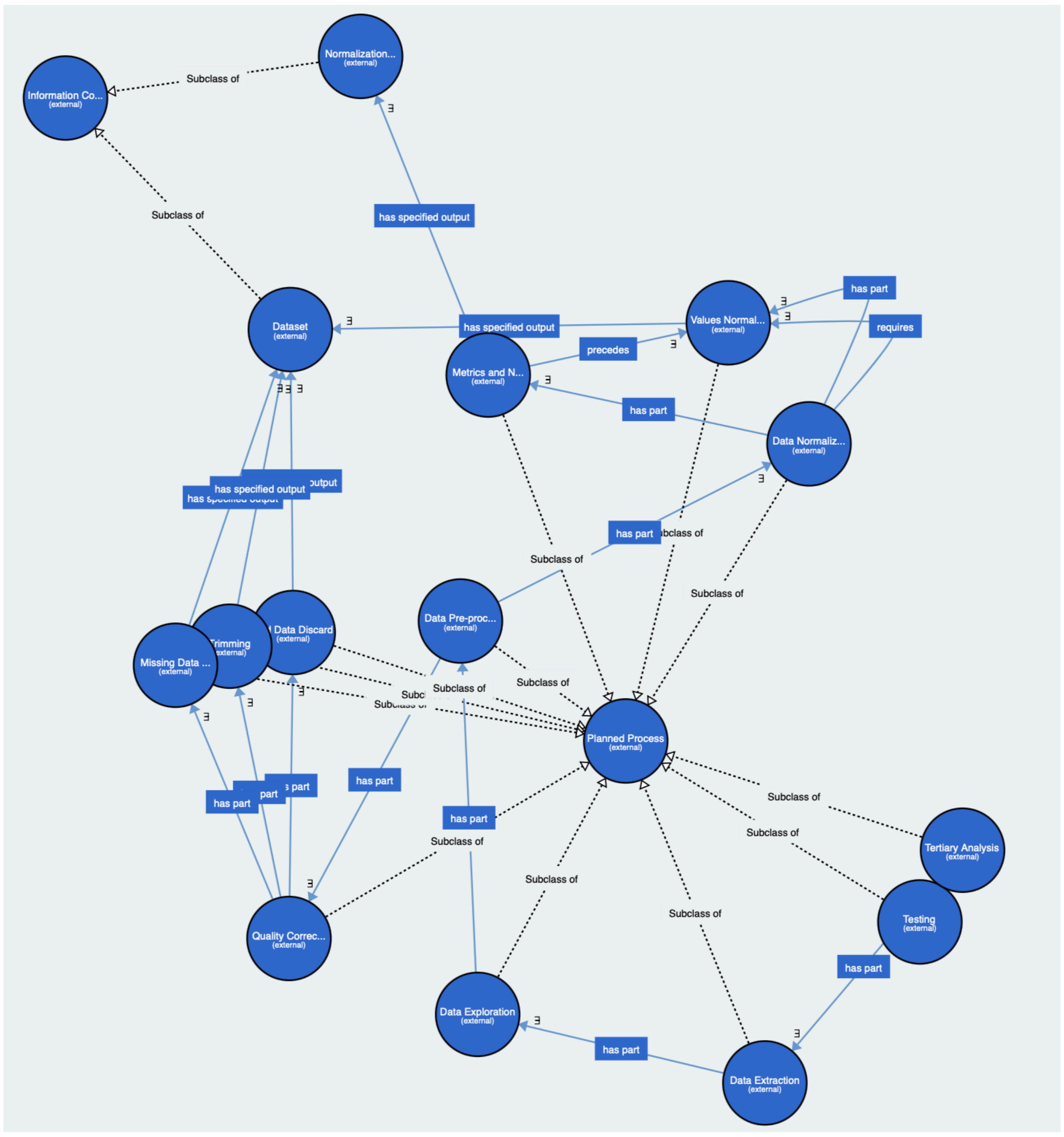

Normalization... (external)
Information Co... (external)
Dataset (external)
Values Normal... (external)
Metrics and N... (external)
Data Normaliz... (external)
Data Pre-proc... (external)
Data Discard (external)
Trimming (external)
Missing Data ... (external)
Planned Process (external)
Tertiary Analysis (external)
Testing (external)
Quality Correc... (external)
Data Exploration (external)
Data Extraction (external)
Subclass of
has specified output
has part
requires
precedes
∃

FIGURE 2.9 – *Zoom on a section of the graphical representation of the ontology provided in Figure 2.8. In particular, it represents the section of the tree comprising the path from Bioinformatics Tertiary Analysis task to Data Preprocessing, with all its children tasks. Source [12]*

3. *Precedes* relation expresses the process's temporal constraint. Both the subject and the object must be Planned Processes. Intuitively, if *A* and *B* are Planned Process and *(A precedes B)* holds, then the output of *A* is necessary for the correct execution of *B*. Precedes relation is transitive: if *A, B*, and *C* are planned processes, if *A precedes B)* and *(B precedes C)*, then *A precedes C*. Finally, the Precedes relation is specified explicitly only between siblings. The following property holds: if *A, AA, B*, and *BB* are planned processes, *(A has part AA)*, *(B has part BB)*, and *(A precedes B)*, then *(AA precedes B)*.

4. *Requires* relation models the necessity of preliminary operations for the execution of the task. Intuitively, if *A* and *B* are Planned Processes and *(B requires A)*, if an instance of the process presents *B*, then it must present *A* as well.

2.3.6 Discussion

Numerous applications and uses are possible for this ontology-based model. As we have shown while verifying the Hierarchical Task Tree using literary works, the first and more apparent one is the potential use of a single terminology to explain procedures. The model provided can be used as a guide for bioinformaticians, who will have a thorough step-by-step explanation of the entire operation. The project can also be coordinated by work groups, who can refer to a schematic diagram of the research process. Software programmers may also use the approach to create new bioinformatics support tools that are easier to use and more suited for inclusion in the research process.

Because of the formalism used, the process model is machine-readable. Tools may use it to describe processes, but they can also use the representation to incorporate process knowledge and use it as a foundation for building operation pipelines or to validate user-requested activities.

Finally, as we will see in the remaining chapter, ontology is a solid starting point for designing user-centered bioinformatics tools.

2.4 GeCoAgent Requirement Analysis

Once the Bioinformatics Tertiary Analysis process had been defined, it was time to understand which functionalities the platform should provide and how the conversational agent could effectively convey them.

2.4.1 Process Requirements

Phases Definition

Starting from the ontology, we decided that GeCo Agent should support the following operations:

- Data Extraction (Data Retrieval and Exploration)

- Data Analysis
- Results Analysis (Computational Results Evaluation)

Figure 2.10 shows on the Hierarchical Task Tree which part of the process is covered by the GeCoAgent. We decided to exclude the Data Integration step because this is an operation that is something that is performed only in complex research pipelines, as shown by the low occurrences of these tasks in the papers surveyed. These pipelines are typically used by expert computer scientists, who are not the primary user of our system. We also excluded the Biological Evaluation in the Result Analysis since it is not a computational step. We arranged all the tasks in the four-step pipeline shown in Figure 2.11.

The researcher should define objectives, including what data should be extracted and how it should be arranged and accessed. Users then establish a universe of interest that may be investigated and assessed in further detail. The resulting universe is examined using various statistical, qualitative, and quantitative visualization tools in this second stage. Data may be evaluated after it has been extracted. The third part entails designing an analytical technique, which entails knowing the suitable procedures/tools, the proper parameters to define, and utilizing a repository of current methodology (such as Statistics, Machine Learning, and Deep Learning libraries) and widely used solutions. With additional visualization tools, the fourth stage of result inspection may be carried out after the result of interest has been created.

As the ontology definition shows, different paths can combine the four phases in many ways. For example, the user may first observe the entire universe to get ideas from all the available data and their interactions, then concentrate on a particular area, or they may choose datasets one at a time and assess their qualities. The user may also decide at any time that the initial datasets no longer meet the analysis requirements and return to the data extraction stage and repeat the procedure. During analysis, methods and parameters are gradually adjusted until the user is confident in the results.

Recurring Pipelines Definition

The next step was identifying recurrent pipelines or frequent sequences of data extraction or processing tasks. Table 2.3 lists illustrative pipelines for these activities. The tasks were chosen to satisfy many requirements while also being orthogonal and hence simple to combine.

We used a top-down strategy regarding data extraction pipelines, building on the expertise we have gained from utilizing the GeCo framework, which is the consequence of our interactions with bioinformaticians who frequently carry out tertiary data management. Examples include combining all experimental areas from several studies into a single sample (file), pairing experimental regions when they are close together, counting experimental regions that overlap a reference dataset (such as genes), and extracting sections that have been confirmed (e.g., regions which overlap in at least a given number of experiments).

Instead, we used a bottom-up approach to data analysis by looking at the analytic pipelines that were used most commonly in tertiary data analysis, which are typically performed

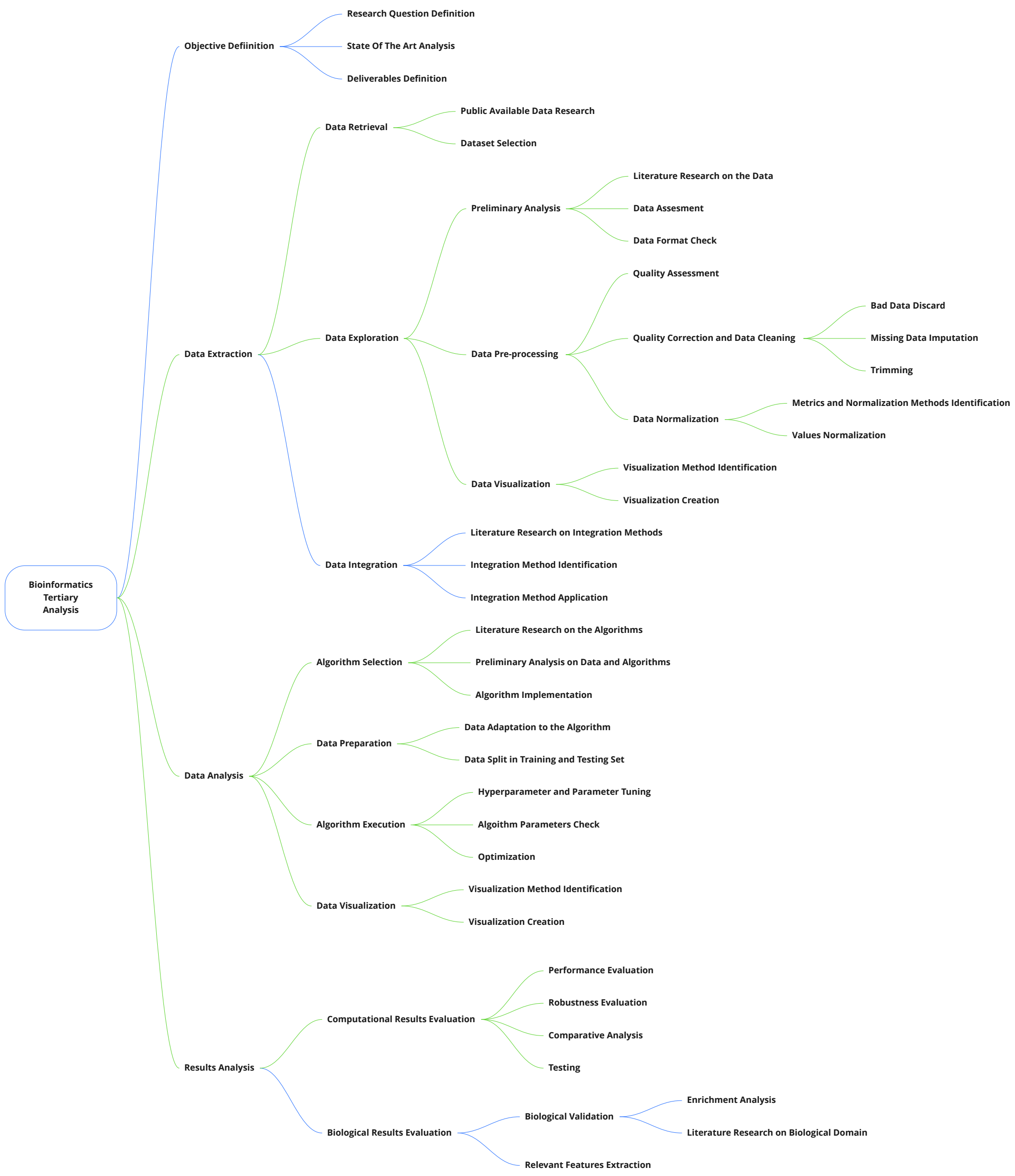

Bioinformatics Tertiary Analysis
Objective Defiinition
Research Question Definition
State Of The Art Analysis
Deliverables Definition
Data Extraction
Data Retrieval
Public Available Data Research
Dataset Selection
Data Exploration
Preliminary Analysis
Literature Research on the Data
Data Assesment
Data Format Check
Data Pre-processing
Quality Assessment
Quality Correction and Data Cleaning
Bad Data Discard
Missing Data Imputation
Trimming
Data Normalization
Metrics and Normalization Methods Identification
Values Normalization
Data Visualization
Visualization Method Identification
Visualization Creation
Data Integration
Literature Research on Integration Methods
Integration Method Identification
Integration Method Application
Data Analysis
Algorithm Selection
Literature Research on the Algorithms
Preliminary Analysis on Data and Algorithms
Algorithm Implementation
Data Preparation
Data Adaptation to the Algorithm
Data Split in Training and Testing Set
Algorithm Execution
Hyperparameter and Parameter Tuning
Algorithm Parameters Check
Optimization
Data Visualization
Visualization Method Identification
Visualization Creation
Results Analysis
Computational Results Evaluation
Performance Evaluation
Robustness Evaluation
Comparative Analysis
Testing
Biological Results Evaluation
Biological Validation
Enrichment Analysis
Literature Research on Biological Domain
Relevant Features Extraction

FIGURE 2.10 – *The sections of trees colored in green represent the operations supported by the GeCoAgent web interface*

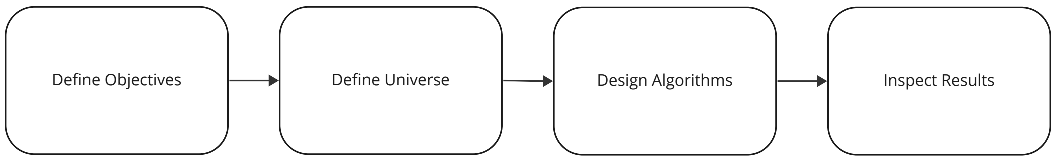


FIGURE 2.11 – *The four interaction phases of GeCoAgent*

Table 2.3 – *Listing of data extraction and analysis pipelines, i.e., recurring task sequences in GeCoAgent processes.*

Extraction Pipelines	**Analysis Pipelines**
E1: Extract relevant regions from a dataset	**A1**: Identify outliers
E2: Merge all regions from multiple samples of a dataset	**A2**: Feature selection or imputation
E3: Pair regions from two datasets at minimum distance	**A3**: Patient/Region Classification
E4: Count experimental regions overlapping with a reference dataset	**A4**: Regression/Survival Analysis
E5: Combine samples from two datasets	**A5**: Network Analysis
E6: Exclude samples from a dataset which intersect another dataset	**A6**: Co-occurrence or mutual exclusion analysis
E7: Extract confirmed regions from a dataset	**A7**: Pattern matching
E8: Count experimental regions overlapping confirmed experiments	**A8**: Mutation Analysis
E9: Build new properties for each region of a dataset	**A9**: Gene Enrichment Analysis

using commercial Python and R packages. We discovered many generic statistics and machine learning tasks, such as feature selection or imputation, that apply to genomic datasets just as they are to any other dataset in data science. We also identified a few domain-specific activities typical of tertiary genomic analysis, including mutation analysis and genomic pattern matching.

2.4.2 Conversational Requirements

We exploited the semi-structured interviews we conducted with 8 researchers described in Section 2.3.2 to understand the desired attributes of the conversational interface of GeCoAgent. In fact, during the interview process, we made participants inquiries about their research objectives during tertiary analysis and progressively delved deeper into a variety of topics, including the line of reasoning researchers would follow to achieve their objectives, the kinds of high-level tasks they would need to complete, the types of questions they would ask themselves at the various steps, how they would formulate such inquiries for a tool like GeCo, and what they would hope the system would provide in response; the process' degree of (non-)linearity, including the times when they would need to“go back” to a prior stage to redo, undo, or alter what they have already done.

We used Thematic Analysis [119] to identify the significant themes by grouping the interview transcripts and sticky notes that the researchers who were being interviewed eventually filled in. We then defined each theme in terms of the conversational requirements for the Conversational Agent and the entire conversational interface, which influenced the design of GeCoAgent as described in the following sections.

- *Process Awareness*. The conversational agent should be familiar with the overall procedure of tertiary analysis; at all times, it should be aware of the step of the process that is currently being performed, and it should use this contextual knowledge to understand the user better, produce more accurate utterances, and suggest proactive next steps (e.g., speaking about the possible supported actions). The discussion is produced dynamically by considering the process environment rather than rigorously pre-defined a priori. Additionally, the conversation is parametric in that the information and activities the Conversational Agent may "talk about" at any time are determined by the process's present context. To orient the user and emphasize the conversation's context-driven character, the Conversational Interface should also make the process' present context clear through textual or visual cues.

- *Discontinuity in the research process.* Tertiary analysis in bioinformatics is a substantially non-linear research process; therefore, when it is carried out, the resulting topology of connections and activities is complex. Goals may be vague or ambiguous initially, and operationalizing them may take multiple back-and-forth stages and trial-and-error scenarios. High-quality analytical data are frequently the result of several little modifications and enhancements made at various stages of the overall process pipeline. As a result, the conversational agent must allow users to return to a prior phase, reverse or change earlier decisions, and ultimately rejoin the process. It should also offer suggestions (see the previous point) that give the user access to various options but may be retracted at any moment.

- *Strong modularity.* The majority of research processes are created by combining the same set of operational stages, which must be executed repeatedly and may do so with various choices of input variables and/or chosen datasets. As a result, the user should be allowed to create and reuse modules as needed. The Conversational Agent should be able to identify this modularization process and use modules as conversational primitives.

- *Multiple dialogue strategies.* Depending on the context (see above) and the user's "at large" goal, the Conversational Agent should be able to sustain a variety of discourse styles and techniques. For instance, the conversational style of the Agent should be that of an executor who interprets user requests for data and operations and translates them into system functions with little dialogic interaction, as would be the case when the researcher has a clear requirement in mind (for example, which could be detected because she has followed a forward flow thus far and expressed precise questions).

- *Multimodality.* Any interface enabling this process contains " data visualization features" and functions on such visualizations, given the visual character of many processes and outputs involved in the tertiary analysis. Additionally, several straightforward tasks may be accomplished more easily by making a few clicks on a traditional GUI instead of asking the Conversational Agent to complete them. As a result, the Conversational Agent functions in a multimodal interaction environment; as such, it must be able to comprehend how conversation-based interactions and GUI-based user interactions interact with one another, as well as how to master the content and flow of discussion in each case.

2.5 Design

We model tertiary genomic research as a series of processes that includes data extraction, exploration, analysis, and result display without losing the ability to be generic. Therefore, a workflow language representing each step and how they relate to one another is best suited to describe the process. This produces a directed graph where each node represents a function or atomic operation, and the edges show how they are related and in what order they should occur. The workflow is formed by first defining the functions and, subsequently, their relationships.

2.5.1 Functions Definition

The most recurrent pipelines, or the repeated high-level processes computed during a tertiary analysis, were specified in Section 2.4.1. Although pipelines are excellent at describing the research process, they are too abstract to define it from an operational standpoint. Pipelines are made up of several processes linked together in a predetermined order rather than being atomic. Additionally, many of the jobs' component processes are repeated across numerous tasks rather than specific to one another. For instance, the selection of the samples is included in both the extraction tasks E1, "Extract relevant areas from a dataset," and E2, "Merge all regions from multiple samples of a dataset."

According to their roles in the four phases of the process, functions can be categorized from a semantic standpoint as follows:

- *Data extraction functions* are used to extract specific data sets from the repository and expand the information obtained using both standard relational algebra operations (including selection, projection, and grouping) and bioinformatics-specific ones (e.g., mapping, binning). Data extraction functions are orthogonal, expressive enough to allow for the creation of arbitrary processes, and brief enough to create a small-footprint workflow that can be created using a conversational agent. In order to do this, we primarily used the semantics of the GMQL operators [25], which operate on GDM datasets and elaborate both genomic data and related information. The Pivot functions are also included in this class; they convert GDM data into flat tables (or Excel files), using arguments to specify which data types must be represented as rows and columns. The Pivot function's output completes the Data Extraction stage and creates tables that may be displayed and utilized in data analysis procedures, or the "Universe of Discourse."
- *Data exploration functions* are used for summarizing and visualizing extracted data in tabular format. Summarization functions enable the computation of various statistics on the data, including grouping, averaging, and counting. Data attributes are shown using visualization tools like charts (e.g., histograms, pie charts, and boxplots).
- *Data analysis functions* can be either domain-specific (such as algorithms that include genomics information) or domain-independent (such as machine learning methods and algorithms, as well as validation functions for gauging the success of

the data analysis). They may be constructed using higher-order composition, as shown in Figure 2.12b, which includes a particular data analysis function encased within a validation method and then encapsulated within a parameter tuning function.

- *Result visualization functions* support the visualization of the results; statistical tests, most frequently employed in scientific data analysis, are also included. These features enable users to convey the study's findings to other scientists and fully comprehend the analysis results.

The supported functions of GeCoAgent are listed in Table 2.4 and are organized according to the phase in which they are employed.

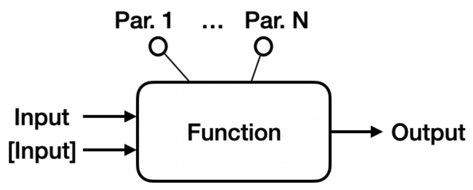
Table 2.4 – *List of functions divided by phase of the analysis process.*

Extraction	**Exploration**	**Analysis**	**Visualization**
GMQL-like functions	**Summarization functions**	**Domain Indep. functions**	heatmap
select samples	min	classification	line chart
select regions	max	clustering	histogram
project metadata	median	regression	PCA diagram
project regions	avg	reduce data dimensionality	piechart
cover	find distribution	complex networks	violin plot
union	density	hypothesis testing	cluster map
difference	summary	parameter tuning	Kaplan-Meier curve
map	**Visualization functions**	**Domain Spec. functions**	enrichment analysis visualization
join	heatmap	mutation analysis	
merge	line chart	motif discovery	
Pivot functions	histogram	enrichment analysis	
pivot	PCA diagram	**Validation**	
label samples	piechart	cross-validation	
join pivot	mutation enrichment visualization	AUC, accuracy, precision, recall, F1	
flat		AIC	
		BIC	

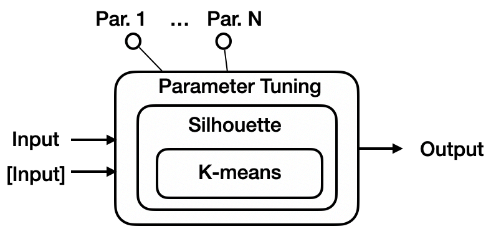
Functions can be *unary* or *binary* from a syntactic perspective, depending on how many inputs they take. The behavior of functions depends on zero, one, or many *input parameters*, which may be optional or necessary. Each function generates a *data output* that may be used as input by the one after it. Additionally, they may have *side-effects* like displaying a visual, recording the analysis, or producing a Jupyter notebook section that encodes the analysis. Figure 2.12a shows a schematic illustration of a general function. Functions are visually represented as nodes of process graphs.

Higher-order functions are a particular class of functions frequently utilized throughout the data analysis stage. Graphically, higher-order functions are depicted using a composed structure in which the higher-order functions encapsulate the parameter functions. Higher-order functions need one or more functions as parameters. One such structure is shown in Figure 2.12b, where "Parameter Tuning" is an example of a higher-order function that takes the "Silhouette" function as a parameter; in turn, "Silhouette" is an example of a higher-order function that takes the "K-means" parameter.

There are two primary benefits to the formulation through the definition of the function. First, we offer a collection of primitives that may be used to define any activity effectively.



(A) *Graphical representation of a function. Each function can have one or more data inputs, one or more parameters, and a single data output.*



(B) *Graphical representation of a higher-order function for data analysis. Parameter tuning is computed over the Silhouette function that, in turn, is applied to K-means, a domain-independent data clustering function.*

Figure 2.12 – *Structure of a simple and high-order function. Source [11]*

For instance, two GMQL functions, "Select samples" and "Merge," are used to complete task E2, which is to "Merge all areas from numerous samples of a dataset." On the other hand, the modular formulation of functions permits their unrestricted combination, even if a specified objective does not require it. In this sense, tasks serve as more of a recommendation for the user than the only means of engagement. Users can choose to utilize the functionalities freely or adhere to these recommendations.

2.5.2 Task-driven Workflow

A genomic application is a procedure that uses functions as nodes and executes them according to a hierarchy of precedence. In computer science, it is customary to visualize processes as directed graphs, where the nodes are the functions, and their arguments and the edges reflect the execution's flow. Figure 2.13 displays a straightforward application that involves clustering the expression data of a specific illness, in this case, kidney renal clear cell carcinoma (KIRC); the GDM dataset is then converted into a standard table using the *pivot* function. We can identify two tasks as patterns of functions in the first phase of data extraction: "Task E1," which is to extract relevant regions from a dataset and is made up of the two functions "Select samples" and "Select regions," and "Task E5"; combine samples from two datasets, which are made up of two "Select samples," one "Select regions," and a "Union," in other words. An overview of the retrieved data is shown during the following step of data exploration. The user then groups the data using the *k-means* technique in data analysis; the number of clusters is then automatically chosen to maximize the silhouette score. The data visualization step completes the procedure when the user manually assesses the outcomes by visualizing the findings of the collected clusters' Principal Component Analysis.

In platforms such as Galaxy[3], Orange[4] or Taverna[5], a bioinformatician needs to be familiar with the functions that are available, how to give the input parameters for those functions,

[3]https://usegalaxy.org
[4]https://orange.biolab.si
[5]https://taverna.incubator.apache.org

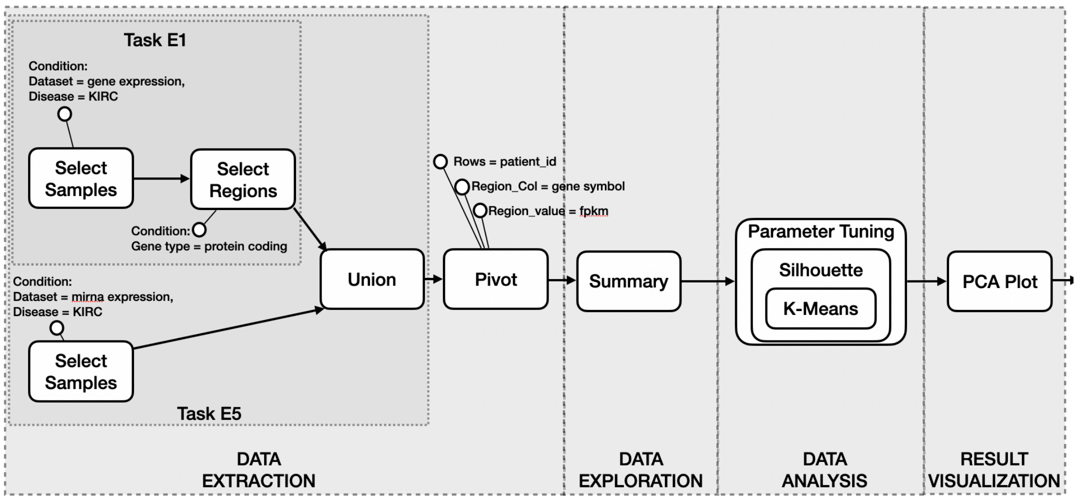


Figure 2.13 – *A function-using program as an example. The user chooses expression information from patients with kidney renal clear cell carcinoma, or KIRC. Then she merges the samples, pivots the dataset, and produces a table with rows for patients, columns for genes or microRNAs, and values for fpkm expression information. To determine the optimal number of clusters, the user plots the data using PCA and then uses k-means coupled with a parameter tuning technique. The results are again plotted using a PCA. Source [11]*

and how to connect those functions with the appropriate workflow edges may create the workflows shown in Figure 2.13. In our context, since we are targeting a biologist or a clinician, they are aware of their needs and intentions but are unaware of the tasks and functions that go along with them. As a result, a conversational agent must capture their needs and intentions through a task-oriented dialogue, and the workflow shown in Figure 2.13 is what comes out of that process. For this reason, we must begin with a redesigned workflow representation that strongly emphasizes user interactions.

2.5.3 Conversation Driven Workflow

Traditional conversational interfaces are based on workflow diagrams, which model the status of a conversation in terms of what the conversational agent understands about the user's needs and intents. During a conversation, the agent is in a specific state of the automaton at any given time and may change that state in response to external inputs (such as messages from the user).

We develop a high-level finite state automaton (shown in Figure 2.14), which can execute any process beginning with the fundamental functions. Every transition in the state machine is a function, whereas the states are the data that result from function executions. All transitions that start in the same state and those that conclude in the same state must accept the same data type and return the same data type.

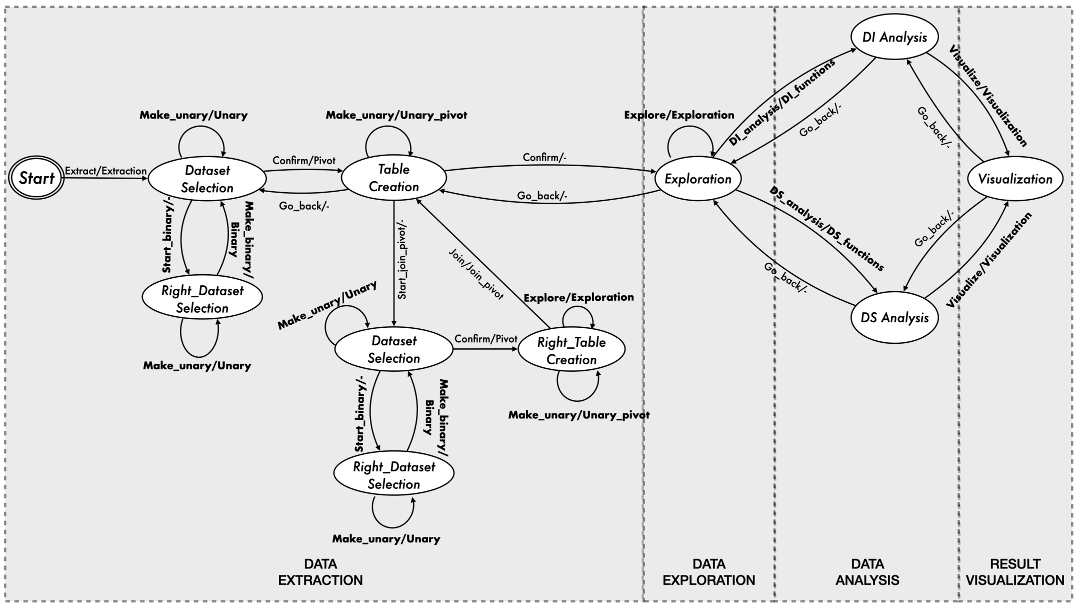


Figure 2.14 – *Finite state machine that describes at a macroscopic level the process supported by GeCoAgent. Every transition represents a function, while every state corresponds to a dataset outcoming from an operation. Source [11]*

For example, the workflow defined in Figure 2.13 is generated by following the path on Figure 2.14 that operates on the `gene expression` dataset by selecting its samples and regions, then adds as right dataset `miRNA expression` by selecting its samples, then performs the binary `union` operation. The confirmation allows the creation of the table using a `pivot`, which is next explored using the `summary`. The analysis starts with domain-independent clustering analysis, performed by the `K-Means` method and `Silhouette` validation, which is repeated multiple times to tune parameters. The visualization enables the display of a PCA diagram at the conclusion. Due to the automaton's generality, users may alter their workflow at any moment to suit their needs in addition to performing predefined activities.

Since it can map the dialogue at the level of functions, we refer to this automaton as the *macro automaton*. This automaton falls short of adequately capturing the dialogic-level interaction. In order to get the intended outcome, functions must be chosen and executed, but they also need to have specific parameters defined; these parameters are frequently determined iteratively, requiring more than one contact with the user. To control the dialogue while the particular function is being performed, we create a *micro automaton* for each function.

An example of Micro automata shown in Figure 2.15 is the one responsible for carrying out the data selection function. The macro automaton's initial edge, marked "Extract/Extraction," causes the micro automata to be activated. The user chooses whether to obtain experimental data or annotations from the outset. In the first instance, three parameters will

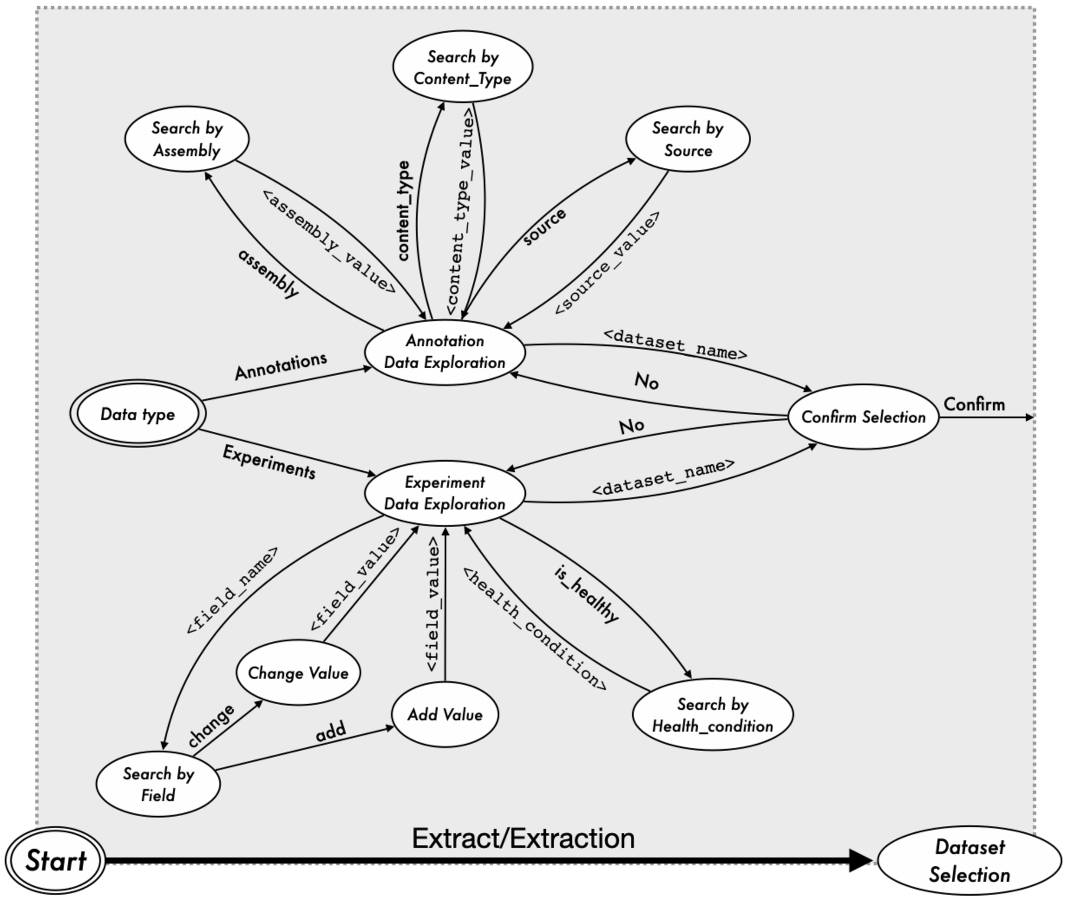


Figure 2.15 – *An illustration of a micro automaton for dataset exploration, triggered by the macro automaton's initial edge with the label "Extract/Extraction." The user's intents and the function's parameters are recorded thanks to the dialog, and information is gathered in the context object. The macro automata start running again when the micro automaton has finished running. Source [11]*

be requested from her: *source*, *content type*, and *assembly*. The second one requires more parameters, such as the *disease* or the *tissue* of the samples. Particular attention is given to the possibility of searching by *health condition*, an essential aspect of many datasets used in human genomics. In the end, if the user confirms the choices, the selected samples are extracted.

2.5.4 From Automata to Conversation

The conversational agent operates generally in the following manner: it begins the execution of the macro automaton by requesting the user to specify the target of the first operation. The conversation seeks to comprehend the user's wants and intents, either directly specified or inferred through responses to context-specific inquiries; when the user selects, GeCoAgent instantiates the appropriate micro automaton, and its execution starts. The automata direct the conversation's generating process: from each state, the agent actively communicates with the user, parses her answer to determine her purpose, and then proceeds to follow the edge designated with that same intent. At the *micro*-level, the technique of querying and understanding the user's responses is used to complete all the necessary dependencies and arguments for a specific function, call it, and update the context. Some of the function's prerequisites may be grabbed straight from the context, missing out on the part of the dialogue and making the agent less verbose.

The agent may foresee potential decisions by utilizing state diagrams to know the exact state of the system. GeCoAgent may thus look at the tasks that are compatible with the trail followed throughout the interaction and proactively recommend the most appropriate functions to novice users. GeCoAgent can ask questions to determine the precise action that the user requires, for instance, once the user has chosen two datasets and needs to conduct a binary operation between them. In conclusion, the following elements help GeCoAgent record a conversation:

- a *macro* automaton, represented in Figure 2.14 that captures the processing of functions; as depicted in the picture, every edge is labeled with a pair of identifiers "intent/micro," where the first label denotes an intent captured by reading the last user's message and the second label, that can be missing, identifies a micro-automaton. For simplicity, similar labels and micro automata are grouped; they are represented using **bold characters**. E.g., **Make_unary** identifies the generic intent of performing a unary operation on the dataset, such as selection, filtering, or grouping, while **Unary** groups the generic micro automata associated with the unary operations. Also, to facilitate the reading, all the trivial edges are not represented, such as help requests;
- a collection of *micro* automata, each of which represents in detail the dialogue for a specific function; the function's needed inputs, outputs, parameters, and connections are extrapolated from the conversation by the micro automata;
- a companion *context object* that stores details about the decisions the user made during the conversation, such as a list of values that have already been supplied for some

parameters and may be used again without requiring a new conversation to be created.

The pre-defined activities in Table 2.3 match the most common interaction patterns; they enable early intention detection and dialogue focusing. Dialogue generation is greatly aided when a pattern is understood, and GeCoAgent can suggest the best default options. For example, during data extraction, after the user has selected mutations, the bot can ask if she is interested in exploring how they distribute to known genes; or during analysis, the bot can ask if the data analysis is concerned with classification or clustering (e.g., about the clustering, validation, and parameter tuning methods).

The conversational engine may leverage the information acquired from the task specification, the interaction history, and the previous executions to suggest the user's best options. In this sense, the conversational agent serves as both an operation executor and a process facilitator who may assist the user activity during the whole procedure.

2.6 Deployment

GeCoAgent's design uses a traditional three-layer architecture and relies on cutting-edge NLP and conversational technologies to interpret the user's intent from textual conversations and a highly expressive multimodal web interface.

2.6.1 Architecture

Today, many open-source tools are available for the automatic construction of conversational agents from a high-level specification of the anatomy of the discourse; the most well-known are Chatterbot, Botpress, and Rasa. These solutions are made for straightforward applications where the interaction is considerably more predictable, such as reserving a table at a restaurant or maintaining a bank's FAQ. These solutions typically do not enable the extensive background activities required for creating and carrying out a complicated data science process and instead limit their features to analyzing user inquiries and predicting the best answer. As a result, we created a standalone software system, whose architecture is shown in Figure 2.16.

The user may pleasantly communicate with GeCoAgent thanks to the *frontend* interface. It is a single-page web application made up of several functional modules, and it was created using the Vue.js framework to provide modularity and extensibility, as is further explained in Section 2.6.4. The backend connection is arranged by the module manager, who also distributes the data that the server sends. A group of separate modules elaborates the information before being shown on the graphical user interface.

The most important components of the backend are: the Natural Language Understanding (NLU) unit, the dialogue manager (which includes the support for the automatons, the context state, the summarizing, and the Jupyter notebook production modules), a relational data engine used to provide quick access to extracted datasets, and the executable

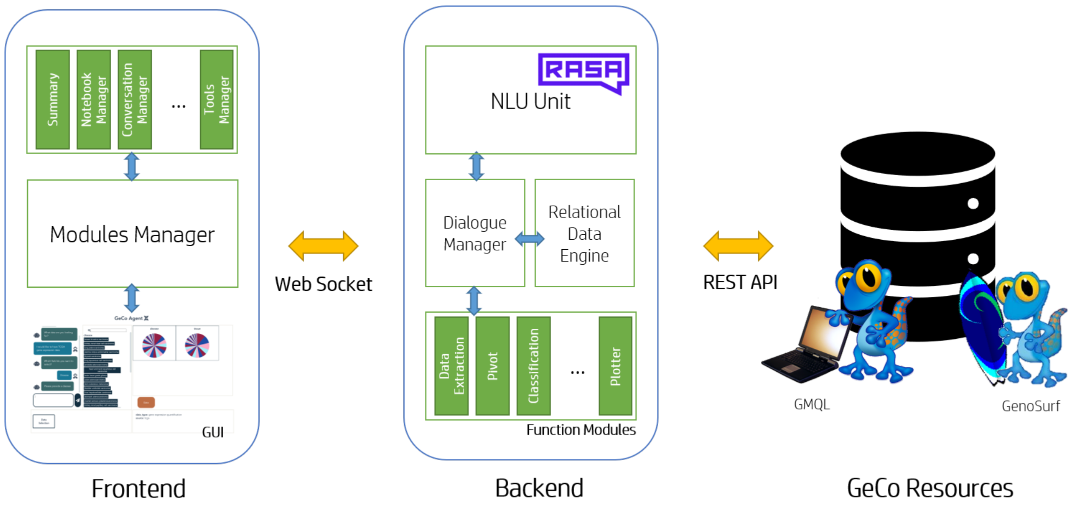


Figure 2.16 – *GeCoAgent's three-tier architecture. The modules manager in the frontend communicates with the server, distributes data to the operational modules, and refreshes the GUI as necessary. The NLU Unit (based on Rasa), the dialogue manager, and the development of the function modules are included in the backend. A relational engine facilitates the quick execution of data exploration and analysis primitives on exported data. GeCo Resources are part of the data layer. Source [11]*

implementation of the aforementioned atomic functions from which any workflow is composed.

The *data layer* is made up of two interrelated parts: GenoSurf and GMQL, the former of which is used to get integrated data that has been curated, and the latter of which is used to manipulate a retrieved dataset while taking into account the genome. The backend calls the REST APIs exposed by both modules to carry out the data extraction operations. The backend's conversation management module also makes frequent calls to GenoSurf APIs to create answer phrases and provide the user with a list of potential field values.

The backend is implemented in Python; it exploits the Flask framework to communicate with the frontend and manage users' sessions. The GenoSurf REST APIs, the GMQL engine, and standard Python techniques for manipulating the tabular data produced after a Pivot are used to construct the data extraction services. Utilizing the large selection of libraries for data analysis and visualization, the inspection, analysis, and visualization functions are fully implemented in Python. At the foundation, a relational data engine with the appropriate materialized views and indexes supports the quick execution over extracted data. Web Sockets are used for the backend and frontend to communicate with one other (using the socket.io package). This makes fast, real-time, bidirectional communication possible; frequently, the backend initiates communication by pushing information to the frontend.

2.6.2 Natural Language Understanding

Software for natural language processing may be created using a variety of chatbot frameworks, such as Google DialogFlow, IBM Watson, and Rasa. Because Rasa NLU [120] offers a comparably more significant number of intents, we chose it as our natural language understanding library. Rasa, in particular, uses machine learning algorithms to comprehend the meaning of words, making it the preferable choice for creating unique NLP for sophisticated chatbots [121].

2.6.3 Dialogue Management

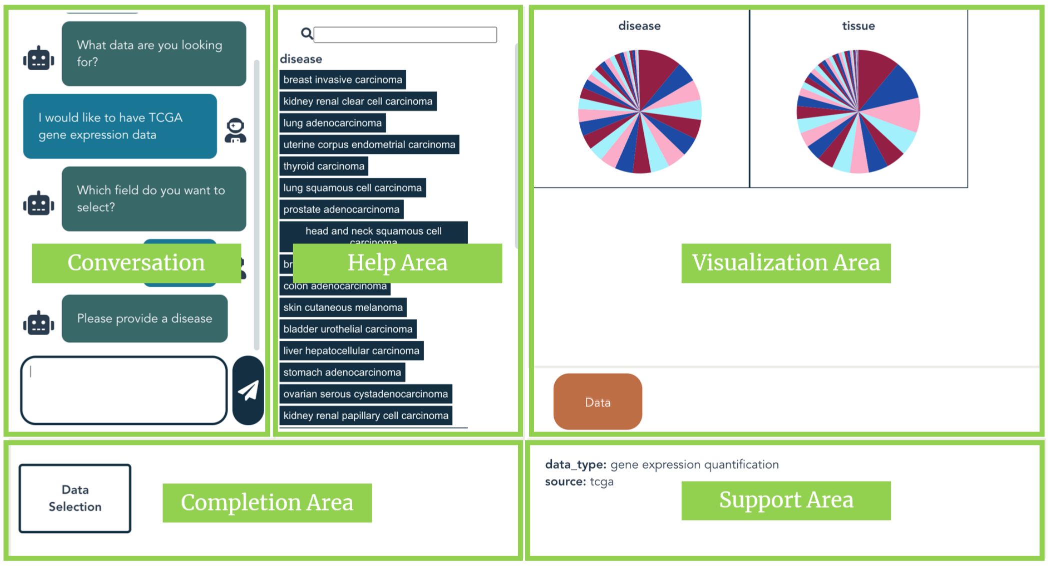
Since the bot constructs the discourse based on the automaton's current state and the context, GeCoAgent's dialogue management module may be considered finite-state and frame-based. GeCoAgent initiates the dialogue, asks questions, and provides the user with options that allow generating the next state. The user has the option to solicit assistance or provide the bot with the information it needs. We also handle problems and allow the user to alter her selections. Changes in options might result in a new workflow distinct from the prior one and alter the dialogue's course. The user can get a summary of the conversation's main crucial messages after the session. The bot offers a Python script detailing the steps taken to collect and analyze genetic data.

2.6.4 Multimodal Interface Design

The interview sessions covered in Section 2.4.2 amply demonstrated that such interaction could not be supported by a conversational interface alone. In reality, when doing their study, scientists must constantly deal with a wide variety of material, including unprocessed data, graphs, tables, documentation, and the past performance of prior procedures. Instead, during a dialogue, only one piece of information may be communicated at a time. Due to these factors, we opted to build a multimodal interface where the discussion is embedded within a conventional GUI. GeCoAgent uses the capabilities of both a conversational agent, which offers ongoing assistance and enables the researcher to work in natural language and a GUI, which offers a channel where many types of information may be displayed simultaneously. Designing the interface, we tried to facilitate at most the executions of the operations that are supported by the tool. This rationale then evolved in a set of guidelines, that will be presented in Chapter 5

As Figure 2.17 shows, the GeCoAgent interface is separated into five functional areas, each of which is assigned to a particular function in the interaction and is arranged in two rows:

- In the upper row we find the *Conversation Area*, the *Support Area* and the *Tool Area*; they describe the tools actively used during the interaction.
- In the lower row, there are the *Process Area* and the *Parameters Area*; they provide information for the user orientation.



What data are you looking for?

I would like to have TCGA gene expression data

Which field do you want to select?

Please provide a disease

disease

breast invasive carcinoma
kidney renal clear cell carcinoma
lung adenocarcinoma
uterine corpus endometrial carcinoma
thyroid carcinoma
lung squamous cell carcinoma
prostate adenocarcinoma
head and neck squamous cell carcinoma
colon adenocarcinoma
skin cutaneous melanoma
bladder urothelial carcinoma
liver hepatocellular carcinoma
stomach adenocarcinoma
ovarian serous cystadenocarcinoma
kidney renal papillary cell carcinoma

data_type: gene expression quantification
source: tcga

Figure 2.17 – *GeCoAgent's web interface. Source [11]*

Conversation Area. The entire conversation history is saved in the panel so that users may explore the panel and remember what has already been done. It also contains the user's interaction with the Conversational Agent.

Support Area. It was created to include any data that may help the researcher throughout the interview. The design reflects a compromise between two demands: on the one hand, the dialogue must be maintained as straightforward as possible, containing just the information pertinent to the current job; on the other hand, the researchers need support data that directs them on what can be done and what cannot. This panel, which is solely dedicated to this purpose, has received all of the material previously in the Conversation Area. The researcher can access two different types of information in the Support Area:

- the options that are accessible at this point in the dialogue, such as the potential actions at a specific stage of the pipeline or the range of values from which to select when applying a filter to a dataset;
- advice and pointers on how to use GeCoAgent most effectively.

A help icon is displayed when a list of options is displayed. A tool-tip panel that details what the user may do with the listed options is displayed when the mouse hovers over the icon. The tool-tip will indicate that parameters can be added (in combination or disjunction with the already inserted ones) or withdrawn, for instance, if researchers are filtering the data and must pick according to which parameter to filter by. The Support Area includes a search bar due to the cardinality of the options set, which in certain situations might be significant (i.e., tens of potential components). Users may explore the information on the interface sequentially as needed, preventing them from being overloaded by everything at once. In conclusion, researchers have three options: they may only use the chat, look at the options displayed by the Support Panel, or get complete assistance through the help tool-tip.

Tools Area. It includes visual feedback for the user that is shown. Since there may be more than one visualization, users may choose which one they want to view by clicking on it in the lower portion of the panel. The user can change between the visualizations, but only one may be displayed at once. According to the conversation's flow, visualizations are dynamically added and deleted; when a new one is created, the associated tag in the lower section is modified. To help the user decide what to concentrate on, GeCoAgent may switch between the visuals. In any case, the user is always free to switch between tabs. The following equipment was found to be required:

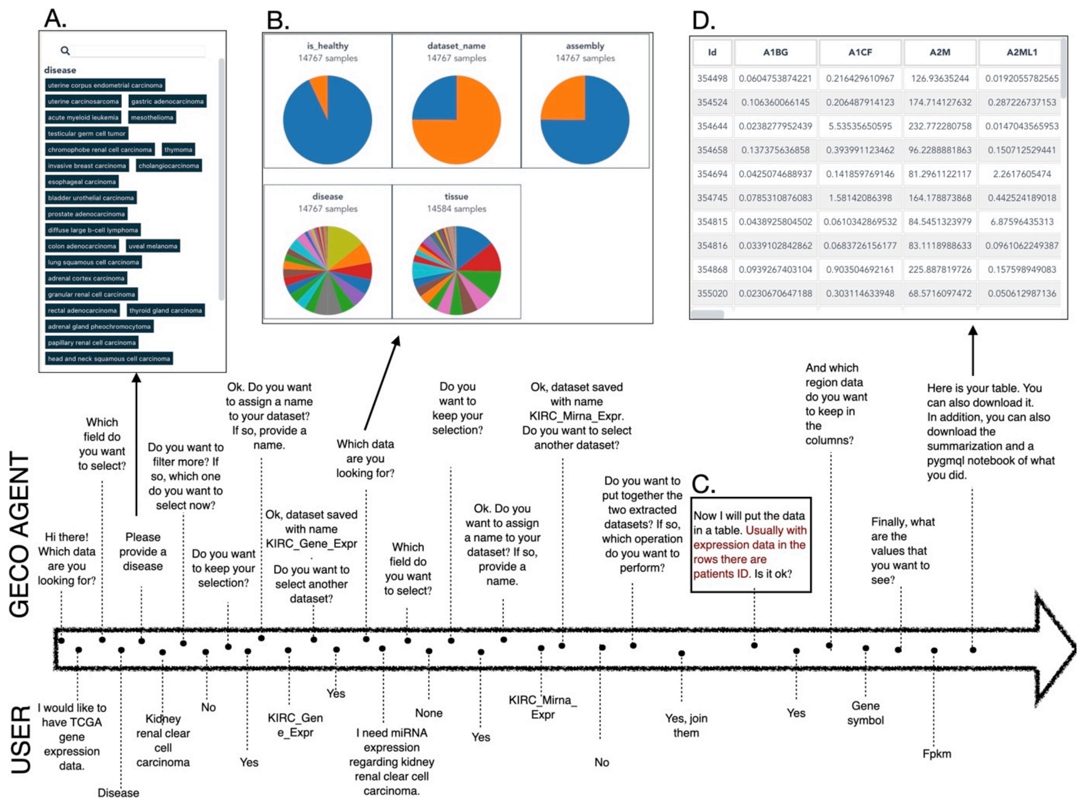
- *Data Visualization*, where graphs and charts are plotted. GeCoAgent automatically generates visualizations, but the user can require additional ones.
- *Table Visualization*, where data tables are shown.
- *Lists Visualization*, where the users can explore lists of any nature, such as information related to the data or sets of manually-curated datasets that are compatible with the operations the user is performing.

Process Area. In addition to giving the user a visual overview of the prior stages, it also gives a visual clue of how the process is progressing by indicating which step is being completed. The panel is made up of many interconnected blocks, each of which represents an action taken at the outset of the execution. The appropriate block changes colors, and a new one is added to the line after a module's execution is complete. A tool-tip panel that lists every decision made in the related module appears when the user hovers the mouse over one of the finished blocks.

Parameters Area. When the execution of the module is finished, the panel is cleaned, and all the same information is relocated to the matching tool-tip in the Process Area. It shows comprehensive information on the current processes.

2.7 Using GeCoAgent: a Concrete Example

We carefully examine a practical scenario to demonstrate how the design decisions manifest themselves in a genuine issue. We imagine that the user wants to receive training in data analysis of expression data. She wants the number of samples available for certain cancers to choose the dataset she uses; to do a statistically meaningful analysis, it is preferable to use a particular tumor with a homogeneous number of tumoral/healthy samples. She wants to arrange the data appropriately for further analysis after retrieving it (e.g., a comma-separated-values file is preferred). She relies on GeCoAgent to help her through the extraction process by supplying information that describes the necessary data since she



Id	A1BG	A1CF	A2M	A2ML1
354498	0.0604753874221	0.216429610967	126.93635244	0.0192055782565
354524	0.106360066145	0.206487914123	174.714127632	0.287226737153
354644	0.0238277952439	5.53535650595	232.772280758	0.0147043565953
354658	0.137375636858	0.393991123462	96.2288881863	0.150712529441
354694	0.0425074688937	0.141859769146	81.2961122117	2.2617605474
354745	0.0785310876083	1.58142086398	164.178873868	0.442524189018
354815	0.0438925804502	0.0610342869532	84.5451323979	6.87596435313
354816	0.0339102842862	0.0683726156177	83.1118988633	0.0961062249387
354868	0.0939267403104	0.903504692161	225.887819726	0.157598949083
355020	0.0230670647188	0.303114633948	68.5716097472	0.050612987136

Hi there! Which data are you looking for?

I would like to have TCGA gene expression data.

Which field do you want to select?

Disease

Please provide a disease

Kidney renal clear cell carcinoma

Do you want to filter more? If so, which one do you want to select now?

No

Do you want to keep your selection?

Yes

Ok. Do you want to assign a name to your dataset? If so, provide a name.

KIRC_Gene_Expr

Ok, dataset saved with name KIRC_Gene_Expr. Do you want to select another dataset?

Yes

Which data are you looking for?

I need miRNA expression regarding kidney renal clear cell carcinoma.

Which field do you want to select?

None

Do you want to keep your selection?

Yes

Ok. Do you want to assign a name to your dataset? If so, provide a name.

KIRC_Mirna_Expr

Ok, dataset saved with name KIRC_Mirna_Expr. Do you want to select another dataset?

No

Do you want to put together the two extracted datasets? If so, which operation do you want to perform?

Yes, join them

Now I will put the data in a table. Usually with expression data in the rows there are patients ID. Is it ok?

Yes

And which region data do you want to keep in the columns?

Gene symbol

Finally, what are the values that you want to see?

Fpkm

Here is your table. You can also download it. In addition, you can also download the summarization and a pygmql notebook of what you did.

Figure 2.18 – *Example of conversation with GeCoAgent. During the interaction, the conversational agent aids the user in completing the task; e.g., (A) the Support Area shows the available options, the Tool Area reports (B) the metadata distribution, and (D) a table representation of the selected dataset. In addition, the Conversational Agent (C) guides the user by suggesting the most common choices or best practices. Source [11]*

is still determining exactly how to pick the correct data and prepare the analysis best. Figure 2.18 illustrates a conversation between GeCoAgent and the user, illustrating the interaction required to help the user obtain the expression data table.

In more detail, GeCoAgent prompts the user to choose the type of data to be used at the beginning of the pipeline. In the beginning, the platform advises that the user identify at least the kind (such as annotations or experimental data) if they do not already have a preference specified. It then moves on to offer the filters that might be used to refine the data further. In the given case, the user needs explicit information about gene expressions. GeCoAgent displays the selected data type in the bottom right panel and a collection of pie charts showing the number of samples available divided by the fields accessible in the top right corner (these, in turn, are shown in the center panel).

The user initially requests to filter on "illness," and after seeing the number of samples in the pie charts, she chooses to extract "gene expression" related to "kidney renal clear cell carcinoma." She continues the talk as though the filtered information satisfies her.

She responds to the bot as instructed, making no changes to her prior selections. After giving the created dataset a name, she may download it by selecting the button in the bottom left panel (this appears after confirming the dataset). Additionally, as it does in this example, the user may choose to use a different dataset. She decides to have both the microRNA expression data and the long protein-coding gene expression data because she wants a full view of the expression data for kidney cancer.

In this situation, GeCoAgent allows one to download the extracted dataset to a local system and give it a unique name. The user wants to terminate the extraction/exploration session by converting the chosen data into a single table containing all the expression values. GeCoAgent offers many functionalities to combine the two datasets (such as UNION and DIFFERENCE).

In this instance, the user chooses to conduct the UNION operation because they want to use the two datasets together. A table might contain distinct information on the rows, different metadata or region data in the columns, and different values presented in the table. GeCoAgent needs specific parameters to turn the data into a table. GeCoAgent offers a proposal to the user, as shown in red text in Figure 2.18, by depending on accepted procedures (i.e., common choice of parameters for certain kinds of data).

In conclusion, the user obtains a table with gene symbols in the columns, patient IDs in the rows, and fragments per kilobase million (fpkm) as values. GeCoAgent offers a file that describes the steps used to get the data, emphasizing the user's decisions and removing conversational aspects that are unimportant for choosing filters and parameters.

```
{ ``Summary": ``You have chosen tcga gene expression
   quantification data with filter 'disease'='kidney
   renal clear cell carcinoma.' You renamed the first
   dataset 'KIRC_Gene_Expr'. You have chosen miRNA
   expression quantification for kidney renal clear cell
    carcinoma. You renamed the second dataset '
   KIRC_Mirna_Expr'. You used the UNION operation and
   renamed the complete dataset 'KIRC_Expr'. The table
   provided has patients ID in the rows, no metadata in
   the columns and gene symbols in the columns, and fpkm
    as values.",
  ``Choices": [
    {
      ``source": ``tcga",
      ``data_type": ``gene expression quantification",
      ``disease": ``kidney renal clear cell carcinoma",
      ``name": ``KIRC_Gene_Expr"
    },
    {
      ``data_type": ``miRNA expression quantification",
      ``disease": ``kidney renal clear cell carcinoma",
      ``name": ``KIRC_Mirna_Expr"
    },
    {
      ``binary_operation": ``union"
    },
    {
      ``rows": ``patient ID",
      ``metadata columns": [],
      ``region columns": ["gene symbol"],
      ``region values": ``fpkm"
    }
  ],
}
```

2.8 Evaluation

In order to evaluate GeCoAgent's efficiency in aiding biologists and bioinformaticians in tertiary analysis, we conducted preliminary empirical research on it. We concentrated our review on *usability*, the primary driving force behind GeCoAgent since our objective was to look into the elements that would either encourage or hinder its adoption within each of the two primary target groups. *Task completion* and *task execution time* are two objective metrics that we gathered to assess how well users performed when completing tertiary analytic tasks. We also considered the subjective data gathered through semi-structured interviews, which were also utilized to interpret the quantitative findings and investigate the *perceived potential for adoption*. The study gave us the opportunity to elicit new needs for our tool because of the insights on typical user behaviors that emerged from our observations of users using GeCoAgent.

2.8.1 Participants

In all, 14 participants from two groups of people (hence referred to as "users") were recruited. Seven biologists with some experience dealing with genetic data made up Group 1, although they had little computer science expertise. Seven members of Group 2 were bioinformaticians or people with advanced computer science abilities and broad computational biology competency but little expertise in gene expression analysis. All users were between 25 and 30 years old (M=26 in both groups), and none had any prior GeCoAgent experience. Three GeCoAgent researchers took part in the study as moderators and observers (collectively referred to as "researchers" in the following text).

Every user signed up voluntarily and was found in clinical or research organizations that had previously worked with our team. They agreed to participate in the study by signing a permission form containing information about the study's goals, methods, and all the legal requirements we adhered to protect the privacy and anonymity of the data we obtained.

2.8.2 Procedure

Due to the pandemic, the study was conducted online using video conferencing software, allowing users and researchers to communicate and observe user behavior while using GeCoAgent. Each participant participated in a single session that had three steps.

Step 1. Presentation of the tertiary analysis activity to be performed. We were curious to investigate our study variables for each of the two primary user profiles—biologists and bioinformaticians. We also wished to contrast GeCoAgent with available tools and/or programming languages (such as Python, R, or comparable) frequently used in bioinformatics studies. Therefore, we distinguished how the two participant groups completed the tasks. Members of *Group 1 (biologists)* only used GeCoAgent to complete the task given. The members of *Group 2 (bioinformaticians)* completed the tasks under two different experimental conditions: *"using GeCoAgent"* and *"using other tools"*, i.e., systems and/or programming languages they were accustomed to in their routine tertiary analytis. Individuals were randomly assigned to the two experimental conditions to counteract learning effects.

The provided assignment included routine tertiary analysis procedures such as data retrieval, data translation into a tabular format, and data clustering. Users had to group information on pancreatic adenocarcinoma patients according to the gene expressions of those individuals. The job was divided into 3 distinct tasks:

- **T1**: find the gene expression data for pancreatic adenocarcinoma; data extraction using GeCoAgent was from GenoSurf [28] (introduced in Section 2.2.4), in "using other tools" condition, users extracted the datasets either from GenoSurf or directly from The Cancer Genome Atlas (TCGA), which is the standard repository for cancer-related gene expression dataset. The two sources contain the same datasets, as GenoSurf imports TCGA data.
- **T2**: create a table to operate on samples, thus with the patients (i.e., samples) on the rows, the gene symbols on the columns, and the expression values (in fpkm) in the cells without adding any labels;
- **T3**: identify the clusters of the patients using k-means clustering, using automatic parameter tuning to find the optimal number of clusters (in the range [2, 5]).

The *task time limit*, or the total amount of time allotted for the completion of a particular assignment, was 15 minutes.

Step 2. Execution of the assigned tasks. During task execution, participants were instructed to "think aloud" [122] or to express any thoughts that came to mind as they prepared for each task, including what they were seeing, thinking, doing, and feeling. Users gave remote researchers access to their displays and turned on their cameras and microphones to see and hear what users were doing and saying. A screenshot of the user's screen during an interview is shown in Figure 2.17. The snapshot depicts the interaction of the

Table 2.5 – *Completion Rate and Time on Task of the participants to the experimental study, divided by experimental condition.*

	Completion Rate			Time on Task					
	T1	T2	T3	T1		T2		T3	
Measure	Total	Total	Total	μ	σ	μ	σ	μ	σ
Group 1 - GeCoAgent	7/7	7/7	7/7	4:45	2:42	2:28	0:41	2:31	0:54
Group 2 - GeCoAgent	7/7	7/7	7/7	2:20	0:49	2:13	1:05	2:23	1:05
Group 2 - Programming Tools	6/7	0/7	5/7	10:21	4:42	N.A.	N.A.	09:38	02:46

user, who first uses natural language sentences while in the second message uses the available keywords, at the point in the dialogue where GeCoAgent proposes to add additional filters and the user wishes to choose the disease.

Step 3. Final interview. The researcher conducted brief semi-structured interviews after each session, emphasizing the perceived usability and usefulness of the tool(s) used to carry out the activities and its potential for adoption. Interviews were captured on video. The inquiries were:

- *(Perceived usability).* How much do you find GeCoAgent easy to use, on a scale from 1 (very difficult) to 5 (very easy)? What are the motivations for your score?
- *(Perceived usefullness).* How much do you find GeCoAgent useful, on a scale from 1 (useless) to 5 (very useful)? What are the motivations for your score?
- Would you recommend GeCoAgent to biologists? For which reasons?
- What are the main advantages of GeCoAgent?
- What are the main disadvantages of GeCoAgent?

Despite not standardized, the first two questions were inspired from some of the SUS ones [123].

2.8.3 Results

Analysis of Quantitative Data

A binary scale (1 = task completed within the time limit; 0 = task not finished by time limit) was used to measure *task completion*. GeCoAgent was used by all 14 users to accomplish the task successfully, as shown in Table 2.5. Instead, T2 was not finished by any user in the G2 group. Four participants made a valiant effort but could not accomplish the task within the given time; three users expressly said they could not execute it and gave up very quickly. Similar behaviors were seen for tasks T1 and T3: one user abandoned task T1 and one user only partially finished task T3.

To compute the *time on task* variable, we only considered the time employed by the users that successfully finished the task in less than 15 minutes, i.e., within the *task time limit*.

The Wilcoxon signed-rank two-paired test comparing the two distributions between those carried out in GeCoAgent (which were finished by all users within the time restriction) revealed that there was no significant difference for the two user groups (T1: $p = 0.219$; T2: $p = 0.579$; T3: $p = 0.579$). Only tasks T1 and T3 were compared for Group 2's time on task in the two experimental circumstances, "using GeCoAgent" and "using other tools." The results demonstrated that utilizing GeCoAgent significantly reduced the time spent on each task compared to "using alternative tools". T1 and T3 were, on average, 3.0 and 3.9 times quicker with GeCoAgent, respectively (Wilcoxon test, $p = 0.0313$, and $p = 0.0156$).

GeCoAgent's *perceived usability* was deemed to be satisfactory based on the quantitative data from the final interview. All users had an average usability score of 4.00/5 (SD=0.78), with biologists achieving a lower average score of 3.86/5 (SD=0.90) and computer scientists receiving a higher average score of 4.14 (SD=0.69). The average utility score for all users was 4.57/5 (SD=0.65), and the value was the same for both groups, with slight differences only in the standard deviation (SD=0.79 vs. SD=0.53). Despite the above data suggesting that some biologists experienced difficulties using GeCoAgent, this did not affect the perceived utility, which was even higher than usability.

Regarding GeCoAgent's potential for adoption, all panelists suggested that biologists employ it. Bioinformaticians did not universally recommended the tool as an alternative to "traditional" programming environments, as highlighted from the answers to the first question in Phase 3; out of the twelve participants, five biologists and seven bioinformaticians did so, while the other two (biologists) did not. GeCoAgent was only suggested by one biologist and two bioinformaticians for the first stages of Tertiary Analysis (e.g., Data Retrieval and Exploration).

Analysis of Qualitative Data

To identify recurring themes in the interview transcripts and thinking-aloud recordings, we employed thematic analysis [119]. In the remaining text, the capital *B* stands for biologists and the capital *C* for bioinformaticians. Even though the system is simple to use, there is an initial learning curve that lasts only for the first exchange of text messages, according to seven participants (4 biologists and 3 bioinformaticians), who also noted that as users have trust in the platform, "*the platform becomes easy*" [B3]. Nine participants *liked* the GeCoAgent chat-based interface. The chatbot questions were defined "*simple*" [C5] and "*easy to understand*" [B1]. C2 enjoyed how the chatbot asked for "*confirmation between one step and the following one*" while C3 found the chatbot proactivity helpful: "*I like how the bot anticipates the steps and suggests them to you.*" Four participants found the visual interface well organized and functional, as "*all the elements are well organized*" and "*on the same interface*", generally providing "*a clear functional organization of all the panels.*" Users praised the graphical interface and conversational interface's seamless interaction. In particular, they liked how the chatbot suggested where to look for information on the interface [C3], how information was displayed synchronously with the chat [B4], and how certain words in the support panel suggested what to do next [B4, C1, C6]. The potential of GeCoAgent as a teaching tool for beginning biologists or bioinformaticians to learn about critical thinking processes and operations to undertake during tertiary analysis was also emphasized by many users in Group 2.

Four computer scientists said that the tool's inadequate openness of decisions, notably the absence of explanations of the parameters automatically employed by the chatbot, was one of its main *drawbacks*. Some users complained that there were times when it was difficult to grasp the options for the next step or that the feedback was too brief in describing what to do; they requested lengthier assistance messages. Additionally, they asserted that they would want "*greater control on the available actions*" [C3] to employ GeCoAgent methodically. Although GeCoAgent's functions are occasionally powerful, biologists have found them somewhat constrained [B1, B5]. They would utilize the tool more frequently if it supported a greater variety of analytical activities.

Lastly, examining video recordings, the researcher notes recorded during the sessions, and conversation logs determined that the chatbot's understanding capacity was sufficient. The conversational agent could not "go back" to a certain step the user had requested (this capability was not yet built). Still, it only twice misunderstood the user's intent. The examination of conversation flows also revealed two recurring patterns of users' conversational behavior: talking to the chatbot "extensively," which means forming whole phrases, and "concisely," which refers to utilizing just the keywords displayed in the visual interface panel.

2.9 Discussion

GeCoAgent's evaluation of a straightforward but practical use case indicates that this tool has a good degree of usability, perceived value, and perceived adoption potential. Despite the many user profiles, there was a short learning curve at first, and participants rated the system as being easy to use. Even though none of them had any prior training or familiarity with the platform, they could execute all the tasks given to them in GeCoAgent.

We believe that by carefully eliciting and modeling the cognitive and operational processes of tertiary analysis, we were able to guide our design approach, which may have helped to make the system's use more predictable and intuitive [124]. GeCoAgent may enable biologists with low computer science abilities to conduct tertiary analytic tasks more successfully and independently, lowering - or maybe eliminating - the need for help from bioinformaticians, filling the gap identified by Bolchini et al. [31]. This is the general conclusion of the study.

However, the use of GeCoAgent might also be advantageous for bioinformaticians. Our technology allows bioinformaticians to conduct some complicated operations more quickly and relieves them of the need to learn new programming abilities, as shown by a comparison of the times required to accomplish tasks T1 and T2 in GeCoAgent and by writing the necessary operations from scratch (which some participants missed, preventing them from completing task T2).

GeCoAgent was initially intended primarily as a tool for experienced genomic analysis specialists. An intriguing finding from the study broadens the range of our target users and the context of use: participants reported that the tool also has potential for novice users as an educational platform, for example, to teach biology and bioinformatics to students and encourage the learning of tertiary analysis.

GeCoAgent's multimodal interaction strategy and the seamless blending of chatbot behaviors with the GUI (Graphical User Interface) emerged as its strongest design elements. The two paradigms were seen as enhancing and reinforcing one another, as previous studies highlighted [125, 126]. However, various restrictions were found in the operational flow that the system provided, whether through the chatbot or the GUI end. In other instances, it was seen to be overly inflexible, missing alternate pathways to reflect the various methods to arrive at a solution or to reflect the user's variable cognitive aim.

Coherently with previous findings (e.g., [127–129]), users would like more adaptivity in GeCoAgent. In fact, some participants employed brief, keyword-based phrases while interacting with the chatbot, and they also valued the succinct tone of the chatbot's utterances. Other people said the chatbot's words needed a more complex style. They would also have required greater feedback on their interactions and more detailed descriptions of the actions to take or the functions carried out by the system. We have a challenging aim for our future research agenda: managing multiple conversational styles and contents to dynamically fulfill the communication needs of the many users on various interactions.

2.9.1 Limitations

The study's findings are generally promising. They are a first attempt to analyze the combination conventional (GUI) interfaces and conversational interaction to support intricate tasks in tertiary analysis, opening the door to the additional study of interactive bioinformatics technology that might result in fresh approaches for researchers working in the area. However, given the exploratory and early character of the research as well as its limitations, the results of our empirical investigation should be interpreted with care. Even while it is equivalent to or larger than earlier empirical research on conversational interaction in related fields, the user sample size is small. For instance, Iris [55] and Ava's [56] empirical assessments comprised 8 and 10 individuals, respectively (using a within-subjects study design). Additionally, we evaluated our system on a limited number of jobs typical of this complicated activity known as tertiary analysis, albeit there are others. GeCoAgent's performance on a broader range of tasks may reveal flaws that the current study could not reveal.

2.10 Conclusions

GeCoAgent is our first attempt at a multimodal conversational agent. It stands out as a system that offers considerable benefits to the state-of-the-art due to the combination of many special features:

- The *GeCoAgent concept* incorporates expertise in genomic computing, both in terms of the high level of abstractions of supported models and languages and the accessibility of reliable systems that offer efficient data management, integrative pipelines, and repositories encompassing the majority of pertinent open data sources.
- *GeCoAgent's requirements* result from a bottom-up discussion with biologists and doctors and a top-down review of experience (both domain-specific and with a broader data science perspective). They enable us to divide data extraction and analysis procedures into alternate phases, each accompanied by exploration and visualization. We identify recurring patterns of interaction within this broad framework, which might be helpful for better process structuring and early identification of user intent.
- The *ontology*, a novel model to describe the bioinformatics tertiary research process. Apart from providing the foundations to design GeCoAgent, the ontology can be used as a reference to describe research activity and to design new, user-centered tools.
- The ontology *elicitation process*, a methodology that can be exploited in any domain to get a deep understanding of how the process workflows are structured and how to support them with innovative digital products.
- *GeCoAgent's architecture* is based on an innovative automata structure that takes into account both the necessity to carry out the standard data extraction and data analysis operations as well as the user's desire. The process' gradual building, which gives dialogues meaning, is another factor that propels the automaton.
- *GeCoAgent's implementation* combines two key technologies for empowering conversational agents: NLP recognition using machine learning acting on a corpus of task-oriented dialogues and a multimodal user interface that combines the chatbot with synchronized data visualizations and with a succinct summary of the developing analysis context.

The output of a design session is delivered in the form of a summary text outlining the entire process as well as a Python script embedded within a Jupiter notebook and linked to both internal and external data resources, which can be used to reproduce the extracted datasets faithfully and statistical results (e.g., Excel tables, diagrams, and plots in standard visualization formats). In this approach, a biologist or clinician may be happy with the user-friendly delivery of findings while also being comfortable that the results can be repeated, run later, and/or used with additional datasets.

GeCoAgent can help to effectively integrate genomics into public healthcare research by bridging the cognitive divide that currently exists between clinicians and biologists regarding the use of bioinformatics tools. This will make it easier to implement personalized medicine in the context of diagnosis, prevention, and treatment. Some of GeCoAgent's benefits are also attributable to the potency of integrative pipelines for genomic data integration, which provide smooth genomic metadata integration and enrichment with innovative biological and clinical research components [27].

From experience with GeCoAgent, we can distill some valuable lessons learned. First, the effectiveness of the platform leads us to think that multimodal conversational interaction might be powerful also in other domains characterized by i) intensively process-driven interaction and ii) high cognitive load for the task execution, paving the ground for more detailed studies in the application of this technology, as we will see in the following chapters.

Chapter 3
DSBot: a multimodal conversational agent for data science

3.1 Introduction and Research Questions

In the previous chapter, we described the work that brought to the design and implementation of GeCoAgent, a conversational interface to support scientists in extracting and analyzing genomic data. The empirical evaluation with users showed the potentialities of such a tool when computational skills are lacking. However, we noticed how the rigid structure of the grammar-based conversation implemented in GeCoAgent requires users to have a little understanding of the operations that are required to produce a meaningful response to their research questions; in fact, GeCoAgent guides users into the formulation of the pipeline but requires users to know the sequence of operation they want to perform, in terms of data extraction and manipulation, and algorithmic choice.

For this reason, we wanted to create a new conversational interface that expanded GeCoAgent in two directions: by relieving users constraints on the domain, operating on generic data science analysis, and users' data science knowledge, automatically translating users' research questions into operative pipelines.

We chose to relieve the domain constraints since whole data science has been an emerging domain in recent years. Businesses use data in the industrial setting to plan predictive maintenance interventions, optimize processes, forecast revenues, and avoid breakdowns [130]. Researchers employ data to support their decisions by validating or developing new ideas [131]. These benefits are amplified by the process of data democratization, in which an increasing number of data repositories are made available online for free access by academics all over the world [132].

However, utilizing the data that is becoming more widely available requires considerable skills in programming, statistics, machine learning, and data management and modeling. As a result, domain experts who may lack strong technical aptitude and computational expertise do not now have full access to Data Science. Activities in data science are frequently tricky, even for expert people. Every dataset is unique and needs a customized set of operations to extract valuable information [133]. Researchers may use an inefficient analysis pipeline or battle with the wrong tools or techniques, frequently leading to inaccurate, if not false, results. Before moving on to more complex studies, they could invest a lot of time in setting up rudimentary data analysis pipelines to examine dataset properties.

Building on the knowledge gained from GeCoAgent, we created the *DSBot*, an interactive machine learning tool based on a multimodal conversational agent that blends Natural Language Processing, conversational technologies, and AutoML methodologies. The goal of DSBot is to convert a research topic communicated in plain language into a data science pipeline that can be executed on any text dataset. The relief of the domain and the users' knowledge constraints implies four design principles on which DSBot bases its foundations:

- **Domain-independence**; DSBot operates on any tabular dataset and is detached from any data repository, making it domain-independent and allowing users to submit their data. Because of this, DSBot can be used for any Data Science application. GeCoAgent, in contrast, is domain-dependent: it only works with its genomic data warehouse and has analysis capabilities tailored to this particular content;

- **Abstraction**; GeCoAgent requires more data science expertise than DSBot, which makes it possible for any domain expert to do complex data analysis tasks on their own data.

- **Declarativeness**; GeCoAgent needs users to supply a procedural specification of the data analysis pipeline or progressively identify the operational steps required to carry out the desired analysis. By expressing the analytic aims rather than the procedures or methods required to produce the desired results, the user of DSBot states their research question declaratively. Instead of specifying the method of obtaining the results, the user may, for instance, query *"What factors influence the price of a property the most?"* to maximize the outcomes, DSBot automatically converts a user's declarative requirements into an operational pipeline by selecting the ideal parameters and methods.

- **Autonomy**; From the standpoint of conversational design, GeCoAgent relies solely on the user's decisions made from a range of pre-defined alternatives presented at each stage by the conversational agent. In DSBot, the discussion is formed by taking into consideration the user's explicit choices and the dataset's characteristics. Both during the elicitation of data analysis needs and the evolution of the pipeline execution, the conversational agent actively engages the researcher in the discussion. For instance, some of the chat is devoted to asking the user if the agent accurately comprehended their wishes. Additionally, DSBot involves the user in key decision points where it is necessary to understand the semantics of the uploaded data and human express decisions. For instance, the question "*Please state the features you want to consider*" or "*Should out-of-range numbers be eliminated because they are probably a measurement error, or should they be considered as acceptable?*" asks what features to choose.

- **Troubleshooting**; When the analysis is complete, the DSBot guides users into a conversational troubleshooting process in which users have the possibility to describe what they do not like about the results (e.g., performance results, how clusters are separated, the number of clusters identified, etc.). In response, DSBot intelligence translates the issue into an actionable solution (e.g., If the users have concerns about how clusters are separated on the graphical representation, DSBbot suggests changing the number of clusters or the dimensionality reduction technique to improve the visualization)

The user can receive help from the beginning (statement of the research topic) to the finish of the data analysis process with the help of DSBot, an end-to-end solution (analysis results reporting). The system analyzes the data and handles pre-processing operations, such as normalizing quantitative variables, addressing missing value issues, or transforming categorical variables with one-hot-encoding representation for clustering analysis, once users have uploaded their dataset and stated their data analysis need (or "research question"). After gathering all the essential data from the user and the data, DSBot uses an automatic machine learning (AutoML) algorithm that was specifically created to choose the best algorithm and adjust its (hyper)parameters. Finally, DSBot delivers graphs and tables that include comments written in plain language and describe the analysis findings.

To evaluate DSBot's ability to convert user information needs into appropriate operational pipelines, we put it to the test on more than 150 "*research questions.*" We also compared the results acquired using TPOT, a well-known AutoML tool, to those produced using the analyses carried out by DSBot over 30 datasets of various types in terms of execution time and outcome. Our findings demonstrate that, with a substantially lower execution time, our system achieves equivalent performances (in terms of accuracy and root mean square error).

Finally, we evaluate DSBot troubleshooting capabilities testing it with 12 researchers with different backgrounds, finding that the totality of the participants could fulfill the task in autonomy, despite most of them have never experienced data science analysis before. In addition, empirical evidence shows that the DSBot troubleshooting system reduced the cognitive effort required to solve data science problems.

DSBot is innovative since it is a new domain-independent tool to help novice users perform data science analyses as well as it is based on a cutting-edge method that combines Conversational Technology, Neural Machine Translation, and AutoML techniques in a sophisticated, original manner. In order to transform declaratively articulated user research questions into operational specifications—i.e., the operations and algorithms that make up the data analysis pipeline—neural machine translation techniques are used. Utilizing concepts and vocabulary that people with little to no expertise in data science can understand, conversation technology is used to engage users in a dialogue devoted to validating with them the accuracy of the operational pipeline concerning their demands. When it is required to elicit more data from the user during the pipeline's execution, conversational technology is also utilized. The "optimal" machine learning algorithm is chosen using autoML approaches by i) running many ML algorithms on portions of the user-uploaded dataset; ii) automatically choosing the best algorithm and the values of its (hyper)parameters; and iii) applying the chosen algorithm to the entire dataset.

Some of the work presented in this chapter have been published in [**pinoli2023ask**, 16, 17].

3.2 State of Art

3.2.1 Automatic Code Generation

Programmers must first understand the programming language before translating their thoughts into it [134, 135], making coding a mentally taxing activity [136]. The goal of a great deal of research has been to create interfaces that convert spoken language into executable code.

Today's automatic code generation tools differ significantly in their operations, the data they accept, and the programming language they output. Shin et al. [137] defined a taxonomy for classifying these applications in two dimensions, the type of the input (a high-level description of the task to be executed or a detailed description of all the commands to be programmed) and the output to produce (executable code, code snippets, or a representation in an intermediate language).

Automatic Code Generation tools can be divided into three primary categories from a technology standpoint. The first one consists of straightforward tools powered by grammar that match natural language patterns and convert them into executable code [138]. The second one entails more sophisticated systems that make use of probabilistic or combinatorial grammars to expand the range of user sentences that are acceptable [139, 140] or that make use of natural language processing to comprehend user requests and glean information that may be used to generate code [141, 142]. The third most recent group uses machine learning methods to create executable programs on demand. In particular, Neural Networks and a sizable training data set are frequently utilized for this purpose [143–145].

3.2.2 AutoML

A subfield of artificial intelligence called Automated Machine Learning (AutoML) intends to automate the machine learning process fully. Data scientists can focus on model optimization and interpretation, and non-machine learning professionals can access machine learning techniques more easily. The most common libraries are AutoML [146]. TPOT, Auto-WEKA, and Auto-Sklearn.

Scikit-learn is the foundation of the AutoML library known as Auto-Sklearn [147]. A meta-learning phase at the beginning of the process to warm-start the bayesian optimizer and an ensemble creation method that integrates models evaluated during the optimization are all factors that help it perform better.

Among the options provided by the Weka platform, Auto-WEKA [148] automatically chooses the optimum algorithm and configuration. The decision is made by converting the algorithm and parameter selection problem into a bayesian optimization problem. Auto-WEKA can select the algorithm and its hyperparameters concurrently or consequentially, depending on the optimization strategy.

TPOT [149] uses genetic programming as an engine for optimization. The tree structures that make up machine learning pipelines are used to perform the genetic algorithm. Each pipeline is assessed, and the best ones are used to build the pipelines that will come after them.

The procedure still requires human input at several crucial stages, such as choosing the proper machine learning problem or identifying the pertinent aspects of domain-specific data, even though automation and efficiency are among AutoML's key strengths [150].

3.2.3 Interactive Machine Learning

With the development of ML and Data Science, there is a growing interest in enhancing Data Science tools to lessen the workload of professional data scientists and to enable sophisticated data analysis for non-experts, hence expanding accessibility and adoption of Data Science solutions. Several studies, like [151] and [152], emphasize the need for machine learning methods and tools that are more interactive and better integrated with human expertise and needs, complementing and enhancing the work of domain experts, especially in situations where providing fully automated functionality is computationally very demanding. We may classify the interactive machine learning platforms that are now available based on how much freedom they give users.

The most straightforward systems enable the execution of a single classification-related machine learning activity. The program automatically conducts the analysis and creates the model; users simply need to provide data with some additional information (such as the label variable, in the case of supervised learning). Ozan [153] proposes using a web interface to build a multi-label image classifier using TensorFlowJS [154]. The system generates a Convolutional Neural Network (CNN) [155] and two files—one providing the network's architecture and the other its weights—by uploading the image files in separate folders for each label.

Google offers a framework called Teachable Machine for building picture and audio classifiers [156]. Users upload samples, and the platform uses a single button click to train a classification algorithm to address the problem without further human intervention. Users can then export the model as a piece of JavaScript code that can be used in any project. Iyer et al. [133] suggest Trinity, a web interface for spatial data analysis that automatically generates binary and multi-class classificators. Data are pre-processed and ready for CNN-based learning, and users are provided with visualizations. If the model's output meets expectations, Trinity provides a method to put it into use.

Other platforms forego complete automation of the process and allow users to compare the platform's suggested solutions to choose which algorithm performs the best. To compare AutoML solutions interactively, Model LineUpper [157], for instance, integrates visuals and Explainable AI approaches. The authors derive a set of principles that can be used in constructing a platform for comparing Data Science models by distilling the findings of an empirical evaluation of the system. The necessity of giving users the opportunity to modify models and making processes transparent so that users can understand exactly what the system performed automatically is emphasized throughout all of the standards.

Other systems aid users in choosing the proper operations to carry out the analysis they want. For instance, Snowcat [158] automatically suggests a list of research questions to address using the data being examined. It trains a series of models and offers an interactive dashboard to examine them based on the problem the user selects. Users for additional analysis can also download the created models.

The AutoDS [159] system automatically suggests ML configurations, preprocesses data, chooses algorithms, executes model training, and then displays the resulting pipeline on a web-based graphical user interface and a notebook-based Python programming interface after data workers have uploaded their dataset. In the publication, 30 professional data scientists participated in an empirical controlled study to explore AutoDS; one group utilized AutoDS while the other operated autonomously. The findings demonstrated that AutoDS increased output, and the models created by the AutoDS team were of higher quality and contained fewer faults. The AutoDS condition still had lower human confidence in the final model. This skepticism is primarily motivated by a lack of complete control over the system. Additionally, 43 percent of individuals said they trusted AutoDS, meaning they had faith in the system and thought it was trustworthy (13 percent disagreed, and 43 percent were neutral). Last but not least, just 10% of participants believed that AutoDS will replace human data scientists, with the remaining 50% remaining neutral.

In order to compare various classifiers while considering model performance, feature space, and model explanation, Meng et al. [160] created a visual technique. ModelWise customizes richly interactive graphics to enable various workflows for model diagnosis, selection, and enhancement.

Many analysis tools focus on providing users with a set of tools to utilize, at the expense of necessitating that users have a solid understanding of the methodology they wish to employ. One interface to work with publicly accessible data on Dataverse repositories is TwoRavens [132, 161]. Users can examine the data they selected and select the statistical approach to evaluate them through a graph-based user interface.

Pyrus is a graphical online modeling platform created for constructing data science pipelines [162]. It is built on the separation of concerns principle, with data scientists implementing block units that carry out data science operations in a dedicated interface and domain experts using a block interface to build pipelines using the units already developed in the system. However, in order to use this platform effectively, users must have a fundamental understanding of data science.

Some studies explore the use of conversational technologies during the data science process. As discussed in the previous chapter, Ava and Iris are two examples. Ava [56] works on a structured process: the conversation predicates on a pre-defined process in which the conversation asks users for the desired operations and parameters. Yet effective, this choice constrains users to use only the modules that fit in the process model. Iris [55] acts as a conversational wrapper for data science operators that allows users to compose their pipelines in freedom. Yet, users must know the modules and their functionalities; the conversational layer does not offer support in composing the operations.

In conclusion, interactive machine learning is a young and active study area. Our literature analysis reveals that users need to be well-versed in the subject matter to trust the outcomes generated by these platforms [159, 163]. With our work, we want to close this gap by offering a tool that does not require highly specialized Data Science knowledge, gives users the freedom to conduct analysis motivated by research questions, and offers sufficient details and justification to increase the user's confidence in the outcomes.

3.2.4 Conversational Troubleshooting

Guided troubleshooting is the practice of supporting users in a certain field in finding solutions to specific issues; typically, the user asks a query, which is then verified to identify the underlying issue and carry out a corrective action [164].

Extraction of the question's aim and specific information is necessary to develop programs that can autonomously identify the correct answer and carry out the necessary actions. There are numerous methods for resolving this issue, and they can be classified into two categories—those that rely on human rule setup and those that learn from data—much like dialogue managers [165].

In comparison to automated learning, rule-based approaches, such as taxonomic case-based reasoning [166] or sophisticated slot-filling algorithms [167], allow for more customization of behavior and rules, such as the integration of other services to boost performance. The breadth of these troubleshooting frameworks is usually somewhat limited because they are typically tailored solutions for a particular application in a single domain [168]. However, managing vast sets of rules is challenging. With a multi-bot system that covers different application domains at the cost of a more complex configuration, Subramaniam et al. [168] attempt to get around this restriction.

By using automatic learning techniques [169], the automated approach avoids manually establishing a complex knowledge base. The difficulty in creating domain-specific training data sets and the wide range in answer quality and consistency provide a problem [170].

3.3 Design Principles

We first introduce the design principles we used, which are multimodality, separation of concerns, and extensibility, before moving on to describe the system. By doing so, we want to clarify why we chose these ideas and show how they are applied in DSBot.

3.3.1 Multimodality

As said in the previous chapters, if a system enables several modes of interaction, including text, images, gestures, and more, it is said to be multimodal [171]. Numerous studies have demonstrated that the response to multimodal stimuli is superior to uni-modal [172, 173]. Additionally, since a web application's graphical user interface serves as its primary point of engagement, we want to assist and intervene as little as possible in that environment rather than requesting that they transfer to a different conversational one [126].

A thorough integration between various modalities supports the understanding of users' wishes. In fact, an automated solution should use the context of the request, namely the information about the pipeline that had been executed, to have a better knowledge of it, just as a human field expert would answer a query regarding the system's state. A customized solution for each unique situation might provide additional information, such as past user behaviors on the graphical interface.

3.3.2 Extensibility

With minor modifications to its underlying structure, an extensible solution enables the addition of new behaviors and modification of the ones already present. As the knowledge base expands, if the industry-specific data is ingrained within the application, introducing changes will necessitate interventions at multiple sites, reevaluating all current rules to determine whether they are affected by the changes, and the extension of the features will become quickly impractical.

We want our system to be extensible mainly in two directions: machine learning experts should easily add a new reference pipeline to the knowledge base and couples of problem-solutions in the troubleshooting phase.

3.3.3 Focus on Tabular Data

We want DSBot to accept tabular data, i.e., tables in which each sample, or table row, contains a tuple of distinct features. This is the standard form of a spreadsheet or the output of a DBMS query. The dataset may contain any number of characteristics and, optionally, a target column.

We decide not to support multi-dimensional data, such as images, audio samples, or temporal series, since these data require ad-hoc preprocessing and different analysis techniques, strongly dependent on the nature of the data itself.

3.4 System Overview

In order to complete an analysis task, DSBot goes through several stages; some of them require interactions with the user, while other phases are fully automatized. To get to the final result, users must go through a series of steps, illustrated in Figure 3.1. At a glance, the steps divide into two main interaction phases:

1. **Pipeline elicitation and execution** (Step 1-7): users upload their datasets and formulate their research questions. DSBot first translates the research questions into operation pipelines, then assesses if the translation is correct, and starts to execute those pipelines, having a conversation with the user when a decision must be taken from a data expert.

2. **Results optimization and conversational troubleshooting** (Step 8-9): Users see the results coming from the execution of their pipeline and can modify the parameters to improve the outcome. DSBot supports them in this process, accepting comments on what users dislike in the proposed solution and proposing actionable suggestions to improve it.

The whole process is illustrated in Figure 3.1 and comprises the following nine main steps:

1. The user specifies the target column and uploads the dataset;
2. In order to infer descriptive properties, such as data types or the presence of missing values, the system performs many conventional analyses on the dataset;
3. The user constructs a research query as a statement in natural language;
4. In a Data Analysis Workflow (DAW) pipeline, the system uses a machine translator to translate the natural language inquiry;
5. In order to make sure the generated DAW phrase matches the user's expectations, a conversational agent engages the user in a discussion; at this stage, DSBot may also use the dialogue to extract additional needs from the user in order to improve or modify the DAW pipeline;
6. The pipeline built in this manner can be supplemented with extra operations to cope with the dataset peculiarities (e.g., treatment of missing data and/or outliers); the confirmed DAW pipeline is compared with a pipeline dictionary, from which the best matching pipeline is selected.
7. The analysis's findings are represented visually.
8. Users can describe the issues they see in the results. The chatbot proposes a solution to those problems, highlighting the parameters to modify on the interface to solve the issue.

3.5 Phase 1

3.5.1 Components

Hereafter, we present and discuss the details of the various components that rule the behavior in the first interaction phase of DSBot and show how they interact.

Data Analysis Workflow Domain-Specific Language

The pipelines for data analysis are encoded in a domain-specific language (DSL) called data analysis workflow (DAW). It is a formal language designed to depict data manipulation and analysis activities flow for (a) interpretation and execution and (b) storage and search in the knowledge base. The dataset descriptions and the list of procedures make up a DAW sentence. The first is a list of terms that describe the major dataset properties, such as `missingValues`, `outliers`, and `zeroVarianceFeatures`, while the second refers to

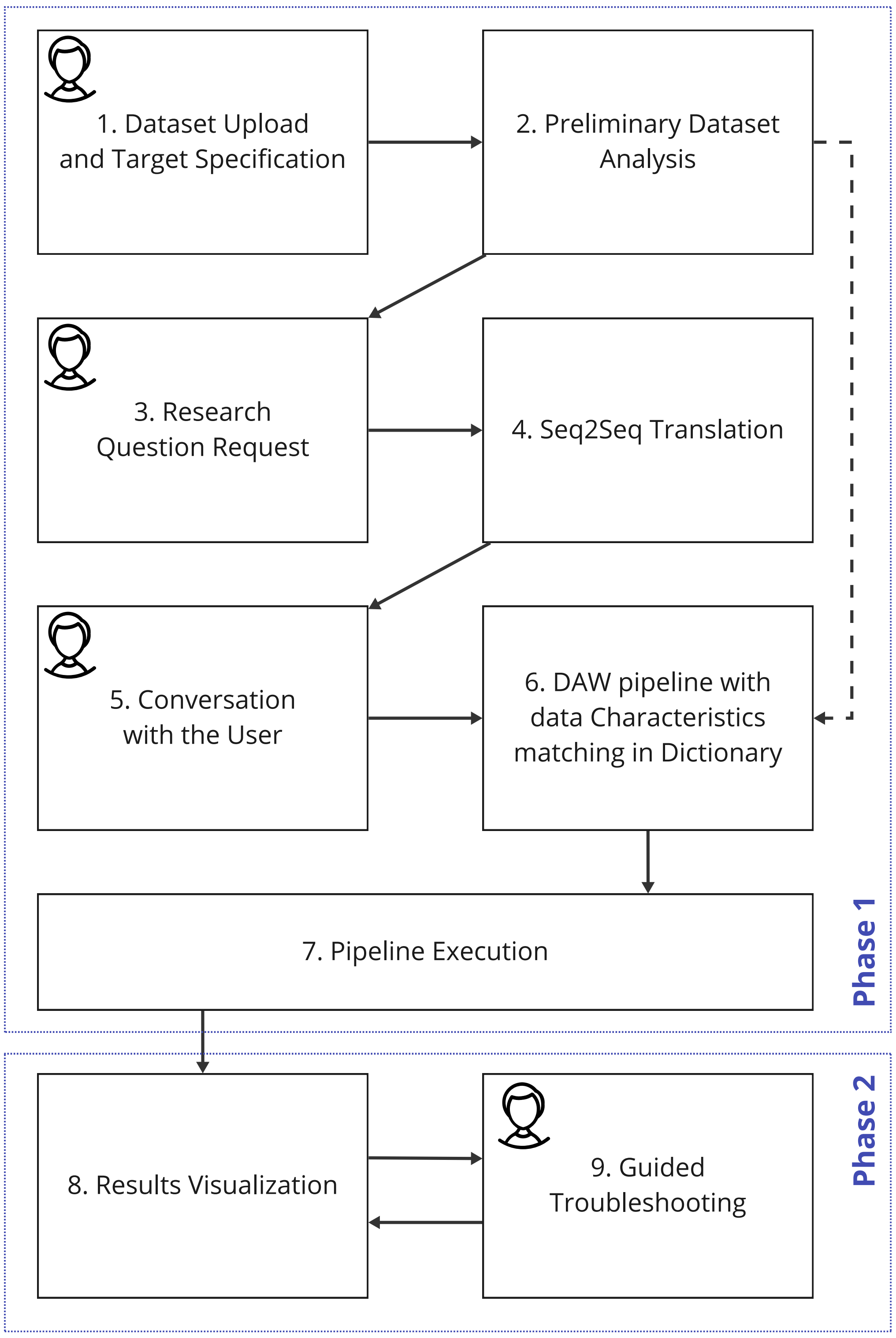

1. Dataset Upload and Target Specification
2. Preliminary Dataset Analysis
3. Research Question Request
4. Seq2Seq Translation
5. Conversation with the User
6. DAW pipeline with data Characteristics matching in Dictionary
7. Pipeline Execution
Phase 1
8. Results Visualization
9. Guided Troubleshooting
Phase 2

Figure 3.1 – *Conceptual architecture of the system. The boxes with a user depicted are the ones in which users must interact with the platform.*

a series of actions that should be taken on the dataset in order to achieve a particular goal. As new capabilities are introduced to the system, new terms can be added to the DAW language's description of datasets and workflow activities.

A workflow description in a DAW is a series of symbols showing the intended processes' linear flow. There are two classes of symbols: *high-level* and *low-level*. Each low-level symbol is a specialization of a high-level symbol. These two classes are arranged in a hierarchy. While high-level symbols must first be specialized in a low-level symbol, either by automatic methods (explained later in this section) or by engaging with the user, low-level symbols have a one-to-one correlation with an operation that may be conducted immediately. Table 3.1 reports a detailed list of the DAW symbols.

High Level	**Low Level**
`missingValues`	`fillMissingValues`
	`removeMissingValues`
	`missingValuesHandle`
`encoding`	`oneHotEncoder`
`outliers`	`outliersRemove`
	`outliersDetection`
`zeroVariance`	`zeroVarRemove`
`strongCorrelatedFeatures`	`correlatedFeaturesRemove`
`featuresToRemove`	`removeFeatures`
`preprocessing`	`standardization`
	`normalization`
`labelOperations`	`labelRemove`
	`labelAppend`
`correlation`	`pearson`
	`spearman`
`classification`	`autoClassification`
	`randomForest`
	`logisticRegression`
	`kNeighbors`
	`adaBoost`
`clustering`	`kmeans`
	`dbscan`
	`agglomerativeClustering`
`outliersDetection`	`outliersDetection`
`featureSelection`	`lasso`
	`selectKBest`
	`laplace`
	`userFeatureSelection`
`featureImportance`	`featureImportance`
`featureEngineering`	`pca2`
	`mds`
`associationRules`	`apriori`

regression	autoRegression
	ridgeRegression
	linearRegression
performance	regressionPerformance
	confusionMatrix
plot	scatterplot
	clustermap
	roc
	lassoPlot
	tableRegression
	tableAssociationRules
	featureImportancePlot

Table 3.1 – *High-level and low-level symbols included in DAW domain-specific language.*

As an example, consider the following DAW sentence:

```
userFeatureSelection oneHotEncode classification roc
```

`userFeatureSelection` is a low-level operator that can be used, and it might need to talk to the user to get more details. Contrarily, `classification` must first be specialized to a low-level operator by a process that will be discussed later because it is a high-level operation. The remaining procedures can be carried out automatically because they are low-level and do not require user interaction.

The pipeline dictionary of the DAW is used to store manually selected models of analyses in addition to describing the analysis that will be conducted. Please note that the DAW symbols represent a particular method and abstract from its parameters, which the pipeline executor automatically adjusts. The language is extensible; thus, new symbols can be added to DAW as new functions are added to the system.

Another advantage of the DAW is that it conceptually and logically separates the creation of the analysis from its execution. Therefore, it would be sufficient to replace the execution engine if better tools were available to do Data Science tasks without compromising the translation apparatus. By simply substituting the existing execution engine with one based on the ML libraries of Apache Spark, one could offer a big-data version of DSBot.

Preliminary Analysis of the dataset

After the user uploads the data and specifies the label, DSBot automatically deduces the properties of the current dataset to guide the selection of potential studies and the selection of an effective pipeline. Table 3.2 displays the list of attributes. Many of these are self-explanatory, but some need more explanation:

Name	**Description**
`missingValues`	Some values are missing or NA
`categorical`	Has both categorical and numeric features
`onlyCategorical`	Has only categorical features
`continuosLabel`	The target has continuous values
`categoricalLabel`	The target has categorical values
`outliers`	Some features presents outliers
`lessThan3Features`	Less than 3 features are present
`strongCorrFeatures`	Presence of strongly correlated features
`uninformativeFeatures`	Some feature is not informative
`zeroVarianceFeatures`	Presence of features with zero variance

Table 3.2 – *Dataset Characteristics inferred by DSBot.*

`Outliers`. This property points out the presence of outliers in the dataset, i.e., data points whose values are distant from the others.

`strongCorrFeatures`. This property highlights the presence in the dataset of columns containing numerical data whose values are strongly correlated, i.e., whose Pearson correlation coefficient is more than 0.9.

`uninformativeFeatures`. This property indicates that the dataset presents one or more categorical columns whose number of distinct elements is greater than half the total number of values.

The system can choose from a variety of pipelines depending on which of the qualities mentioned above best suit the uploaded dataset. Each characteristic is handled by a distinct operation found in the pipelines that are available within the dictionary, as described in Section 3.5.1.

Question Translation

Word embeddings are numerical vectors representing each word by one of the vector's components. Two vectors are comparable if their corresponding points are close to one another in the vectorial space, where vectors are represented as points. These representations, often referred to as dispersed word representations, may extract words' semantic and syntactic details from a sizable unlabeled corpus [174].

Word embeddings have been demonstrated to enhance natural language interpretation and processing tasks. In this study, we used GloVe pre-trained embeddings that use common English language [175]. We changed them into word2vec vectors [176] and included the embeddings of terms typically used to describe jobs involving data analysis. To do that, we first used the Beautiful Soup library, and Google search results to web scrape the contents, including the domain-specific terms. We calculated the embeddings of the new words from the collected corpus using the word2vec technique and a skipgram model, with minimum count and maximum distance being equal to 2 and 30, respectively.

We developed a sequence-to-sequence machine translation model based on recurrent neural networks (RNN) to translate the user's query into a workflow instance to be matched to the pipeline vocabulary. The conditional probability $p(y|x)$ of translating a source sentence $x_1, ..., x_n$ into a target sentence $y_1, ..., y_m$ is modeled by a neural network in a neural machine translation system.

We utilized the OpenNMT tool suite [177] to create the seq2seq machine translator. In order to evaluate its performance, we used around 25,000 phrases for validation after training the model using a synthetic dataset made up of a set of manually prepared templates. After 10,000 steps, the validation accuracy reached 78%.

Conversational Agent for Assessing Comprehension

A conversational agent evaluates if the system correctly understood the input sentence after converting the user's request into a DAW. The system gets the DAW phrase and transforms its actions into a textual description to accomplish this. In order to get the user's approval, descriptions are combined into a single text message. The text defines operations at a high level, removing itself from the specifics of how they work in favor of anticipated outcomes. Data Science jargon is avoided since users of DSBot might not understand it.

We can use the same textual description for many terms that belong to the same algorithmic family since we are more concerned with the result of the operation than the algorithm's activity. The following description, for instance, can be transformed to include both "`kmeans`" and "`agglomerativeClustering`" modules: "*to organize your data in such a way that objects in the same group (called a cluster) are more comparable (in some sense) to each other than to those in other groups (clusters)*". The term "`clustering`" is associated with the same definition.

As a result, the DAW's symbols are divided between high-level and low-level classes. Every symbol in a high-level class has a corresponding symbol or symbols in a low-level class. For instance, the low level symbols "`randomForest`," "`logisticRegression`," "`auto-Classification`," etc., relate to the high level symbol "`classification`."

Every node in the tree can have a textual description that includes the conversational sentence to be used. When a word needs to be translated, the system uses a tree search to find the deepest node in the path from the root to the searched node that has a textual description. It then returns that description, combined with the descriptions of the other words in the DAW, and sent to the user for confirmation.

Users can check the written description, request more in-depth explanations, or request an example of the workflow in action to see if they have grasped it correctly. The same guidelines that guide the creation of texts for explanations and examples also apply here. Users who approve the workflow hand off control to the Workflow Enrichment module (Sec. 3.5.1).

The conversational agent aids the user in selecting an operation if the system has not accurately comprehended what the user wants to do, adhering to the state-machine-based description of the conversation flow depicted in Figure 3.2. Diamond shapes represent the agent's decisions regarding dataset properties, and rectangles with smooth angles represent decisions regarding the data science pipeline that will be suggested to the user when the conversational agent sends a message to the user through the chat and waits for one of the responses shown on the exiting arrows.

The dialogue aims to discover the user's operational goal or the high-level operation they want to carry out, such as clustering, regression, classification, association rules, or correlation matrix. In order to enhance the user experience and aid in comprehension, the discussion uses the dataset information. If the dataset has a label, the prediction of a value is the first suggested operation; if the user responds positively, the system automatically chooses whether to use regression or classification based on the label's nature. Instead, suppose the user has not specified a label. In that case, the conversational agents will initially inquire whether the user wishes to perform tasks such as clustering, prediction, or association rules or correlations to uncover linkages in the data.

Heuristics on the data are used to determine the algorithm to use once the family of algorithms has been found. When the user looks for relationships in the data, for instance, correlation is automatically chosen if the dataset only contains numerical variables, while it is eliminated if the dataset has no other numerical variables. The type of variable to be predicted influences whether classification or regression is used in prediction tasks. Once the desired operation has been elicited, the control is sent to the Workflow Enrichment module, and a new pipeline containing the operation is created.

Pipeline Dictionary and Workflow Enrichment

As shown in Figure 3.1, the DAW obtained as a translation of the research question is compared to a dictionary comprising manually edited pipelines and the dataset's properties. Using known best practices in data science, the best match is utilized to enhance and fix the DAW. For instance, the best match will include the `zeroVarianceRemoval` step if the dataset contains columns with zero variance (i.e., constant values). The pipeline dictionary currently has 9634 pipelines spread across 439 different combinations of pipeline attributes. The pipeline vocabulary can be expanded with new pipelines by design.

This algorithm will choose the optimum matching pipeline based on the dataset's properties as well as the user's input, enhancing the accuracy of the analysis. Figure 3.3 reports an example of analysis results before and after the pipeline enrichment using the example of a user uploading the Penguin dataset and requesting a clustering analysis. The outcomes of the pipeline execution without taking into account the dataset features, specifically `removeMissingValues`, `oneHotEncoder` and `kmeans`, are depicted in Figure 3.3(a).

The outcomes of the second approach, which considers the dataset's features and can extract more important clusters, are displayed in Figure 3.3(b).

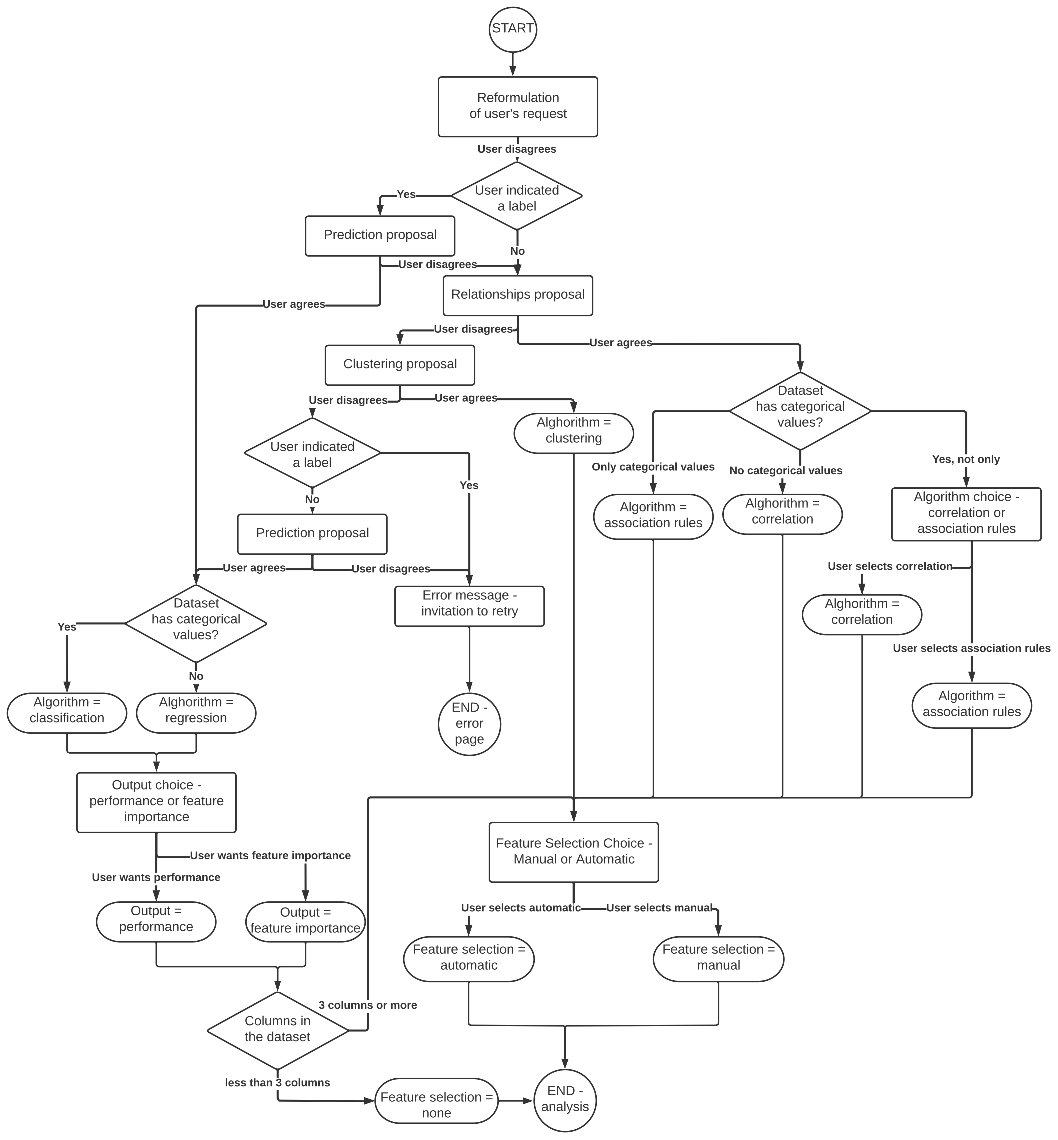
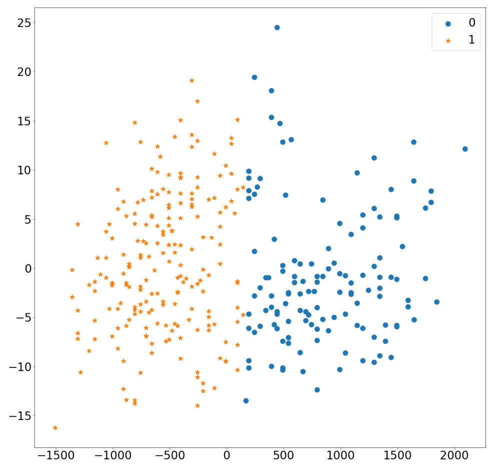
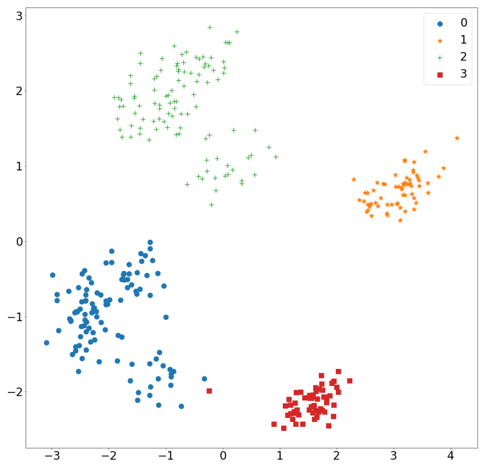


Figure 3.2 – *Finite State Machine of the high-level conversation flow for user's operational goal elicitation.*



(A) *Results of using a pipeline without taking the features of the dataset into account.*



(B) *Results of the pipeline's execution using the chosen workflow while accounting for the dataset's properties.*

Figure 3.3 – *Example of an analysis' results before (a) and after (b) the enrichment of the pipeline.*

AutoClassification and AutoRegression Modules

Not all of the modules in DSBot require user input. Fully automated processes exist; the most pertinent ones are `IRAutoClassification` and `IRAutoRegression`. These two perform several regression and classification modules while adjusting the settings. The module and parameters with the best accuracy and root mean square error (RMSE), respectively, are chosen for the analysis and prediction.

More specifically, the auto-classification module is used following several preprocessing steps and the selection of Lasso features. A Random Forest classifier, an Ada Boost classifier, a k-nearest neighbors (KNN) classifier, and a Logistic Regression classifier are all used to classify once the dataset has been divided into training and test sets. It also runs a parameter tuning module on each of them to compare the four modules with the best combinations of parameters in order to determine which module is the best. For the four modules, in particular, it uses a random search approach based on each module's unique parameters. The search begins by evaluating each candidate (i.e., parameter combination) with a small number of samples and then iteratively chooses the optimum parameter combination using progressively more samples.

The AutoClassification module determines the ideal combination based on the accuracy acquired and runs it on the training set after computing the accuracy for each module and various sets of parameters. It then stores the prediction of the testing sets and the significance of the features so that it can display the performance, a ROC curve, a confusion matrix, or the significance of the features in two separate plots, depending on the user's preference.

We also developed a module to perform auto-regression after some pre-processing steps and a Lasso feature selection. A Random Forest Regressor, an Ada Boost Regressor, a Linear Regressor, and a Ridge Regressor are among the four alternative regression modules performed on the dataset after it has been split into a training and test set. It runs not only these four modules but also a parameter tuning module on each of them to try the best parameter combination and compare the four modules with the best parameters to determine which module is the best for that dataset. The parameter was also tweaked using a random search with cross-validation in the regression scenario. This method considers all possible parameter combinations on a small subset of samples, iteratively chooses the optimal combination, and then applies it to a more extensive collection of samples.

In the regression situation, the parameter was also adjusted using a random search with cross-validation. This method considers every conceivable parameter combination on a limited group of examples. It then iteratively selects the best combination and applies it to a more extensive collection of samples.

The `AutoRegression` module determines the best combination based on the obtained root mean squared error and runs it on the training set after computing the accuracy for each module and various sets of parameters. The prediction of the testing sets and the significance of the characteristics are then saved in two distinct plots that demonstrate the performance and significance of the features, respectively, along with a brief explanation of the findings.

The Flask framework is used to serve the frontend and maintain user sessions in the backend, developed in Python. With the help of the many libraries that are readily available, the analysis and visualization functions are fully implemented in Python and sent to the frontend as pictures. Web Sockets are used for the backend and frontend to communicate with each other (using the socket.io package). This makes fast, real-time, bidirectional communication possible; frequently, the backend initiates communication by pushing information to the frontend.

3.5.2 Architecture

The architecture of Figure 3.4 displays the key elements of part of the system responsible for the execution of Phase 1.

The user can pleasantly interact with the tool thanks to the *frontend*. It is a single-page web application that features separate modules for the web chat, input gathering, and result display. The Vue.js framework was used in its implementation to provide modularity and extendability.

The query translator, the pipeline dictionary, the pipeline executor, and the conversation manager are among the backend's most crucial parts. The Backend operates as follows:

- The *pipeline executor* does the initial analysis on the data after receiving the dataset from the backend.
- The DAW pipeline is created from the research question by the *query translator*.

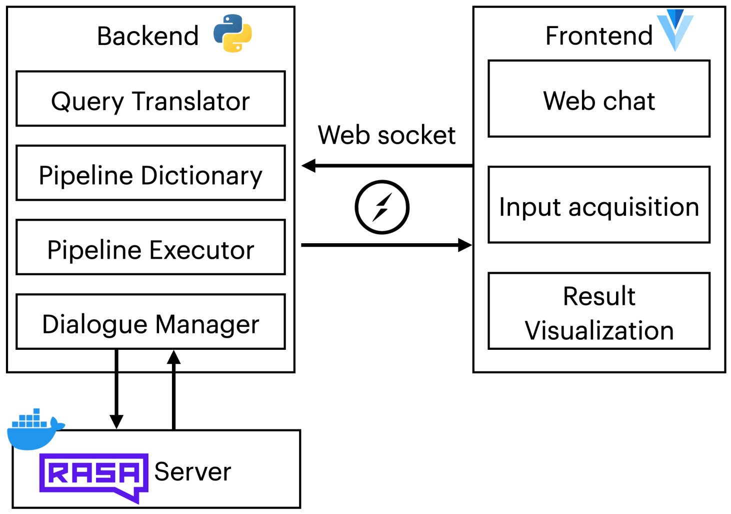


Figure 3.4 – *Architecture of DSBot responsible for the first phase of the interaction.*

- If the translation is accurate, the *dialogue manager* verifies it with the user in a brief conversation and assists the user if necessary.
- The *pipeline executor* searches the pipeline dictionary for the best pipeline, builds a module for each action it needs to perform, and then launches the module. The pipeline executor may need human assistance to finish the pipeline and request specific input from the user. Other processes, such as the ones described in Section 3.5.1 (`IRAutoClassification` and `IRAutoRegression`), instead carry out the analysis automatically. The pipeline executor may occasionally alert the user when doing certain operations by giving her access to important details highlighted while the activity is being performed. The percentage of deleted outliers serves as an illustration.

A Docker instance of the open-source Natural Language Understanding (NLU) Unit RASA [120] interprets users' messages. This service is in charge of translating user dialogue into symbols that are understandable from the dialogue manager (intents) and extracting the parameter required for task completion (entities).

3.6 Phase 2

3.6.1 Problem Definition

We wish to describe the system's fundamental components and formalize the issue mathematically before moving on to the description of the second phase. High-level abstraction demands that the proposed system consider the following: read the user's inquiry and context details into the input fields, consult a knowledge base, then respond with a statement outlining the pertinent details the user needs to be informed of.

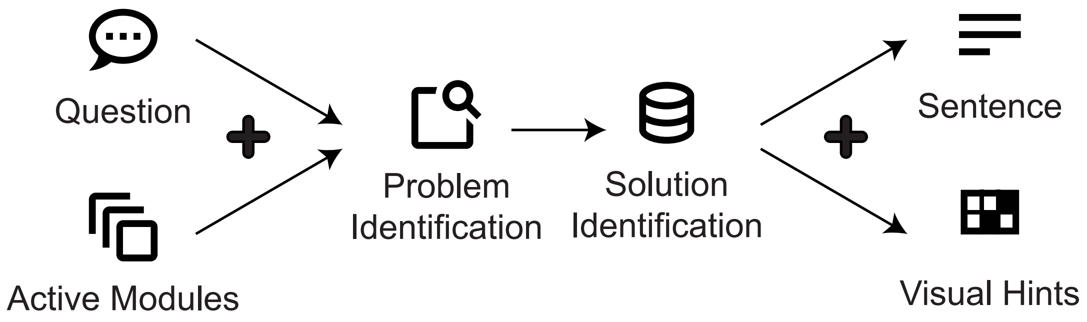


Figure 3.5 – *Sequence of operations performed by the system.*

Defining Modules and Parameters

We can consider the user's options and organize them into modules to describe the context. Whether the user has access to the modules will determine whether they are active. For instance, the modules of a photo editing program can be different tools like the brush, crop, or stamp tool. The settings for each module can also be represented by one or more *parameters*. These could be the number of clusters for the clustering method or the error function in the regression module. The system will then be informed of which modules are currently in use, as inactive modules are not accessible to users and cannot be used to address their problems.

Mathematical Formulation

The issue can be formalized in the following way in terms of mathematics. Assuming that each parameter is linked to one and only one module, let M be the set of all the system's modules and P be the set of parameters. Let q represent the user's query. Our definition of the system, given A and a collection of active modules, $A \subseteq M$, is the relation $\sigma(A, q) = \langle \overline{M}, \overline{P}, \mathbf{r} \rangle$ that, given in input the users' question and the active modules, returns:

- the set of modules $\overline{M} \subseteq A$ and parameters $\overline{P} \subseteq P$ to highlight;
- the textual response **r** that explains to users how what to do to try to resolve their issues

3.6.2 System's Overview

The Configuration Table, a data structure that depicts relationships between issues and solutions in a particular field of application and whose aspect is provided in Section 3.6.3, is the core element of this module of DSBot. Figure 3.5 illustrates the four processes that make up DSBot's action in this phase. With the active modules on the screen—in the example, these are the active tools—the system receives a problem from the interface that the user describes in the conversation, for example: "How do I make the image darker?" Through purpose extraction, the problem is found in the sentence. The list of potential fixes for the active modules is retrieved using this in the Configuration Table.

3.6.3 Configuration

We suggest giving a quick rundown of the data models and configuration files before going into further detail about the system's structure. The *training*, *configuration table*, and *utterances* files are the three files that can be used to configure this module.

`training.json`

The purpose of this file is to train an NLU engine to identify the problem type from the queries that users ask. It contains examples of user statements for each type of problem. The format adheres to NLP.js's standards[1], shown in the table below. Many languages can be supported by offering numerous files with the language locale name—for instance, `training-en`, `training-it`.

This is an example of how the file is structured:

```
{
    ``name": ``training",
    ``locale": ``en-US",
    ``data": [
        {
            ``Intent": ``clusters_not_separated",
            ``utterances": [
                ``Points of different groups are mixed",
                ``I don't see the groups",
                ...
            ]
        },
        ...
    ]
}
```

`configuration-table.csv`

The relationships between the different problem categories, modules, parameters, and solutions are shown in this table, which is shown in Table 3.3. The table-based configuration strikes a balance between extensibility, which is crucial when adding new problem kinds and solutions to the table, and maintainability, which is vital while reading and changing the table.

`utterances.json`

Utterance file is the data structure that links the responses' identifiers to their textual equivalents. It is a straightforward JSON file with the following stated structure. As before, several files—for instance, `utterances-en`, `utterances-it`—can be included with the

[1]www.github.com/axa-group/nlp.js

Module	Parameter	problem1	problem2	problemN
moduleA	param1	utternace1	utterance2	
moduleA	param2	utternace3		utterance4
moduleB	param3			utterance5

Table 3.3 – *Organization of the Configuration Table. Each row represents a module parameter, and each column represents a problem that DSBot seeks to solve. If there is an utterance identifier in a cell, the parameter for that utterance could be used to solve the issue in that column. This table is directly editable by the conversation designer, who can add problems, edit utterances, and establish new relationships between problems and parameters.*

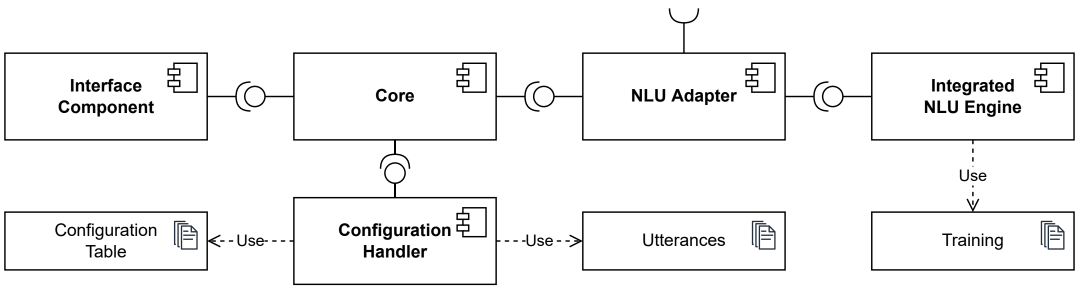


Figure 3.6 – *The components of the system and the configuration files.*

locale name to accommodate various languages.

```
{
``clusters.number":
    ``Try to increase the number of clusters, ...",
``visualization.technique":
    ``If you don't like how your points are grouped, ...",
``clusters.separation":
    ``If you don't see a clear division between clusters, ...",
...
}
```

3.6.4 Architecture

The architecture of the module responsible for Phase 2 is structured in five units, as presented in Figure 3.6. The module is embedded in the frontend; therefore, it is programmed in Vue.js[2], to be compatible with the DSBot application. We decided to create a self-standing component such that we could release it as a multimodal conversational troubleshooting framework to be integrated into other web applications.

[2]www.vuejs.org

The developer can access this element, which may be integrated into the online application. A circle-shaped overlay button is visible at the bottom right of the screen. The button enables the chat panel, which manages communication with the conversational agent, to be toggled on and off. The interface will send a message to the *Core* component when a user sends one, and when it receives a reply, it will display it in the chat and update the information on which modules and parameters should be highlighted in the application. The Vue.js component receives many configuration arguments, such as the URLs of the configuration files and a list of the currently active modules, and fires an event if the highlighted components change.

Core

The Core carries out the module's logic. Information regarding the active modules is provided at initialization. It uses the *NLU Adapter* and the *Configuration Handler* to get the response—which includes an answer and a list of elements to highlight in the interface—after receiving the user's query. The algorithm's specifics are provided in Section 3.6.5. The *interface component* receives the response back. This component, independent of the interface, is implemented in JavaScript.

NLU Adapter and Integrated NLU Engine

The task of extracting the intent, which symbolizes the user's problem, from the user's inquiry falls to the NLU Adapter. It does this by using a lightweight NLP.js instance called the *Integrated NLU Engine*. Due to its browser compatibility and native capability for more than 40 languages, we adopted NLP.js. The web application has been trained with the Training examples and is prepared to extract the intentions from the user sentences when loaded in the client browser.

Configuration Handler

The Configuration Handler then gets the utterances and configuration table documents, parses them, and gives the Core component their contents. The papers can be delivered directly as strings or through a URL.

3.6.5 Runtime Behaviour

The mathematical notation covered in the preceding section can represent the Core component's behavior. The NLU Adapter derives the intent, which stands for the problem type t, from the user's question, q. Assume that A represents the same set of active modules. The method reads the configuration table, where the columns contain the problem kinds tj and the associated cell is designated with uij, signifying an utterance identification, and identifies all the $\langle m_i, p_i, u_{ij} \rangle$ that fulfill $t_j = t$ and $m_i \in A$. In other words, it locates all the current module parameters related to the issue and the accompanying statements. The actual utterances that make up the answer are retrieved from the utterances document using the input language extracted from the user sentence. These are then combined to create a single final response, denoted by the symbol $\mathbf{r}$.

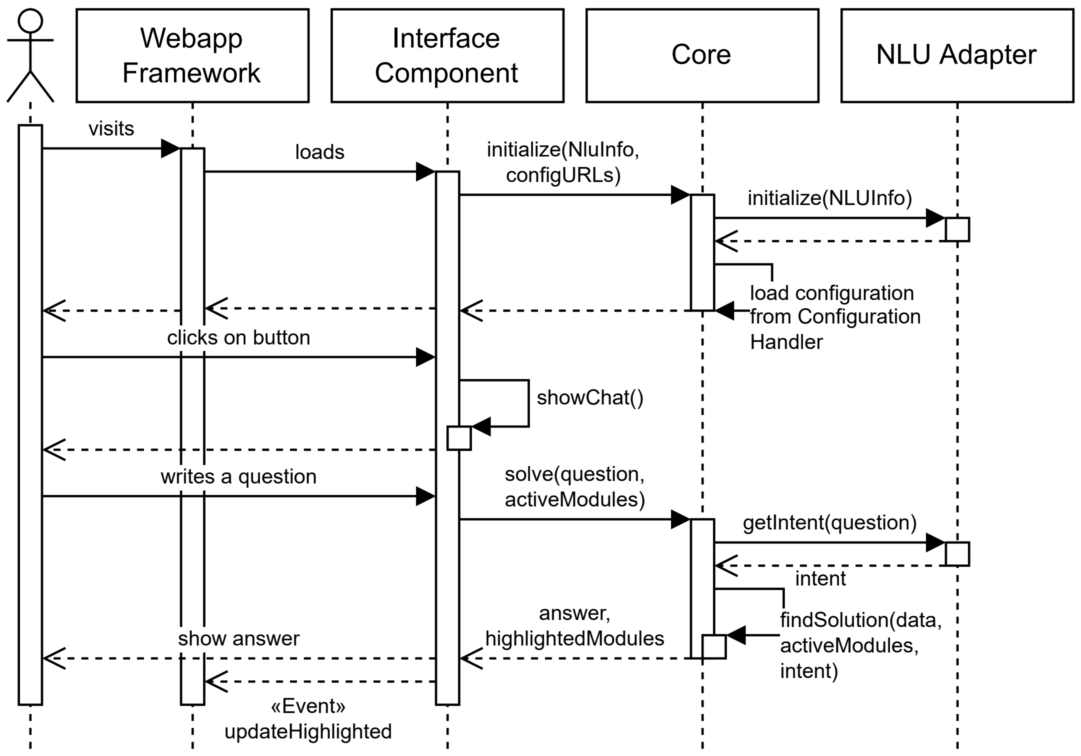

Webapp Framework
Interface Component
Core
NLU Adapter
visits
loads
initialize(NluInfo, configURLs)
initialize(NLUInfo)
load configuration from Configuration Handler
clicks on button
showChat()
writes a question
solve(question, activeModules)
getIntent(question)
intent
findSolution(data, activeModules, intent)
answer, highlightedModules
show answer
«Event» updateHighlighted

Figure 3.7 – *The system's behavior; the Configuration Handler is not shown for simplicity.*

As shown in Figure 3.7, which depicts the user's behavior from the moment the troubleshooting phase is initialized, the user's interactions with the system can be modeled practically. If used, the Integrated NLU Engine is swiftly taught during the early phase. The user can communicate with the system and enquire. The computer reads the question, keeps track of the running modules, and generates a reply. When the action is finished, an event is released to inform the rest of the application that the highlighted items need to be updated. Responding to this event and modifying the user interface is the responsibility of the rest of the system.

3.6.6 Mapping of the Design Principles in the System

All the design principles are respected in the design of DSBOt, even with different strategies in the two phases of the interaction.

In Phase 1, multimodality is declined in a series of uni-modal interactions. The conversation alternates with the interaction on the GUI according to the task users have to accomplish. In the second phase, instead, DSBot will allow intrinsic multimodality. The configuration will be based on a Configuration Table that combines the context—here defined as what is now displayed on the screen—with the problem-solution pairs and enables incorporating in the response the interface elements pertinent to the solution, along with the answer. As a result, various perspectives could generate various recommendations suited to the screen's action.

The suggested system meets the criteria for extensibility. Adding new pipelines to Phase 1 is as simple as adding new lines to the corresponding file. New data science modules can be easily integrated into the system only by enriching the DSL of the new symbols. In Phase 2, a new problem type can be added by adding a column, much like adding a module or a parameter is as simple as adding a row to the table. The use of identifiers rather than placing the utterances themselves in the cell allows for the reuse of the same utterance multiple times without duplication and support for multiple languages; these operations do not require interventions on the code, only on the data structure and on the NLU configuration, fulfilling the separation of co-operation requirements.

The third principle, focus on tabular data, is guaranteed by the system's design since it is the only type of data that DSBot accepts.

3.7 Use Cases

We demonstrate two DSBot executions in this section, one on a dataset of genomic features and the other on a clinical and demographic information dataset. The first one displays an entire analytic example, from uploading the dataset to visualizing the results; the second, which concentrates on the conversational aspect, displays a dialogue in which DSBot first struggles to comprehend the user's natural language request.

3.7.1 Use Case A: Analysis Use Case

In this use case, breast cancer patients' genomic data—one of the most prevalent tumor types—are analyzed using data-driven methods. *Basal*, *luminal A*, *luminal B*, and *her2* are the four molecular subtypes of breast cancer that are most frequently identified [178]. The progression of the disease and the selection of the most effective treatment are both influenced by various subtypes [178].

To determine whether any breast cancer subtypes are easily confused, we assume that the user will be a clinician who examines a genomic dataset containing gene expressions (i.e., the level at which each gene is active within a biosample) for a cohort of 1,127 patients affected by breast cancer. Thus, in our dataset, the columns represent the genes and the rows of the patients. We assess each patient's expression of the 50 genes from the PAM50 panel, which oncologists have determined to be most associated with the subtype of breast cancer. Each patient is also given her subtype on the label.

The web interface for uploading the dataset ('`pam50_m...fed.csv`') with the label 'Expert subtype' is shown in Figure 3.8. In this stage, the user provides the dataset's three properties: the label, the presence of column names, and the existence of an index column. After analyzing the dataset and extracting the necessary attributes, DSBot displays a preview of the uploaded table. In particular, this genomic dataset contains outliers and has a categorical classification. According to the user's query, the best pipeline is matched using these factors.

The user then uses the interface depicted in Figure 3.9 to express a research query in natural language. In the given an example, the user seeks to identify the most challenging breast cancer subtypes. His or her inquiry, which is depicted in Figure 3.9, would be something like this

Can you tell me which are the most similar and the most difficult subtypes to discern?

The user's request is interpreted by DSBot, which chooses the following preliminary DAW pipeline as being suitable:

```
classification confusionMatrix
```

The chatbot asks for confirmation to continue after rephrasing the user's request in line with the identified preliminary pipeline and offering a brief explanation to help the user understand how their request has been interpreted (Figure 3.10 - right side). The inferred dataset features are utilized as input for matching the pipeline dictionary and the confirmed preliminary DAW pipeline. The user is shown the final results after selecting and running the DAW pipeline (described below), shown on the left side of Figure 3.10.

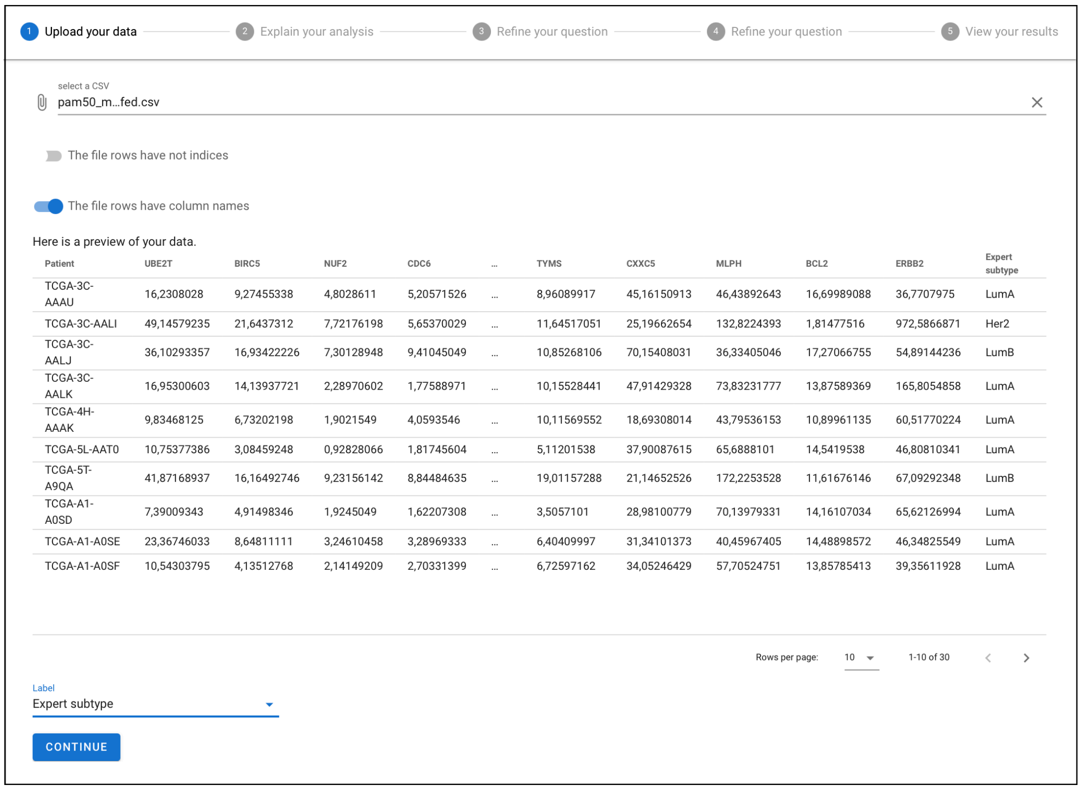
```
labelRemove standardization outliersRemove
lasso autoClassification confusionMatrix
```

The user receives feedback from DSbot while the final pipeline is being executed in the form of sentences like "Outliers make up 2.838% of the rows. I'll get rid of them" (Figure 3.10 - right side). The chatbot presents the main findings when the final results have been visualized.

3.7.2 Use Case B: Conversation Use Case

In the second use case, the user wants to examine the clinical and demographic data from the stroke prediction dataset [179]. Table 3.4 gives an example of a conversation between the DSBot (B) and the user (U) that can occur if the system is unable to understand the user's first request. The conversational DSBot agent makes recommendations for potential alternate analyses while considering the dataset's features. Since the user indicated a label, DSBot suggests performing a prediction analysis. Additionally, since DSBot recognizes that the label is categorical, it suggests using a classification approach rather than a regression one. It then asks whether the necessary analysis should present the performances or the significance of the traits.

Before using the classification algorithm, DSBot advises performing a feature selection analysis; specifically, it requests either an automatic or a human feature selection. The user chooses to manually choose the features and offers a list of options to eliminate. The user must provide some information to DSBot to complete the analysis, such as eliminating or filling in the missing variables. The conversational agent also offers various analysis-related insights, such as the proportion of outliers that were eliminated.



1 Upload your data — 2 Explain your analysis — 3 Refine your question — 4 Refine your question — 5 View your results

select a CSV
pam50_m...fed.csv

The file rows have not indices

The file rows have column names

Here is a preview of your data.

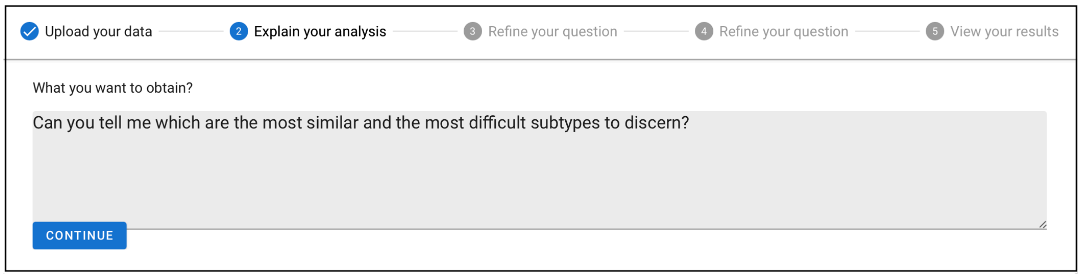
Patient	UBE2T	BIRC5	NUF2	CDC6	...	TYMS	CXXC5	MLPH	BCL2	ERBB2	Expert subtype
TCGA-3C-AAAU	16,2308028	9,27455338	4,8028611	5,20571526	...	8,96089917	45,16150913	46,43892643	16,69989088	36,7707975	LumA
TCGA-3C-AALI	49,14579235	21,6437312	7,72176198	5,65370029	...	11,64517051	25,19662654	132,8224393	1,81477516	972,5866871	Her2
TCGA-3C-AALJ	36,10293357	16,93422226	7,30128948	9,41045049	...	10,85268106	70,15408031	36,33405046	17,27066755	54,89144236	LumB
TCGA-3C-AALK	16,95300603	14,13937721	2,28970602	1,77588971	...	10,15528441	47,91429328	73,83231777	13,87589369	165,8054858	LumA
TCGA-4H-AAAK	9,83468125	6,73202198	1,9021549	4,0593546	...	10,11569552	18,69308014	43,79536153	10,89961135	60,51770224	LumA
TCGA-5L-AAT0	10,75377386	3,08459248	0,92828066	1,81745604	...	5,11201538	37,90087615	65,6888101	14,5419538	46,80810341	LumA
TCGA-5T-A9QA	41,87168937	16,16492746	9,23156142	8,84484635	...	19,01157288	21,14652526	172,2253528	11,61676146	67,09292348	LumB
TCGA-A1-A0SD	7,39009343	4,91498346	1,9245049	1,62207308	...	3,5057101	28,98100779	70,13979331	14,16107034	65,62126994	LumA
TCGA-A1-A0SE	23,36746033	8,64811111	3,24610458	3,28969333	...	6,40409997	31,34101373	40,45967405	14,48898572	46,34825549	LumA
TCGA-A1-A0SF	10,54303795	4,13512768	2,14149209	2,70331399	...	6,72597162	34,05246429	57,70524751	13,85785413	39,35611928	LumA

Rows per page: 10 1-10 of 30

Label
Expert subtype

CONTINUE

Figure 3.8 – *DSBot web interface for adding the input dataset and labeling it (use case A).*



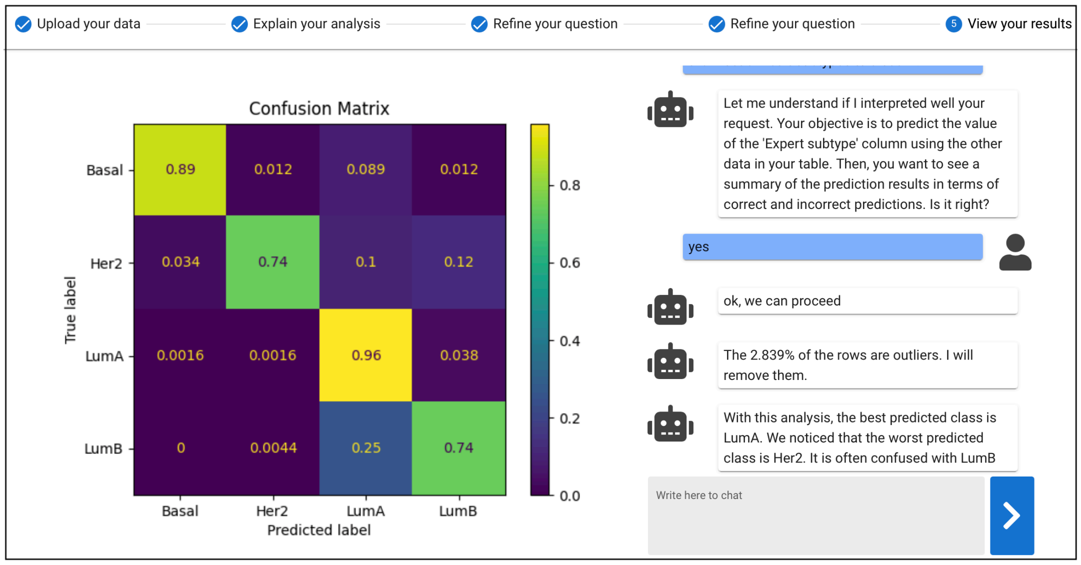
Upload your data — 2 Explain your analysis — 3 Refine your question — 4 Refine your question — 5 View your results

What you want to obtain?

Can you tell me which are the most similar and the most difficult subtypes to discern?

CONTINUE

Figure 3.9 – *Web user interface: textbox for users' research questions (use case A).*



Let me understand if I interpreted well your request. Your objective is to predict the value of the 'Expert subtype' column using the other data in your table. Then, you want to see a summary of the prediction results in terms of correct and incorrect predictions. Is it right?

yes

ok, we can proceed

The 2.839% of the rows are outliers. I will remove them.

With this analysis, the best predicted class is LumA. We noticed that the worst predicted class is Her2. It is often confused with LumB

Figure 3.10 – *Web interface for visualization of final results and chatbot explanations (use case A).*

The reported discussion also highlights the tool's domain independence; in fact, it only considers feature attributes while interacting with the user and choosing the optimum pipeline, ignoring feature semantics.

BOT: What do you want to obtain?

USER: analyze my data

B: Let me understand if I interpreted well your request. First, you want to group your data in such a way that objects in the same group (called a cluster) are more similar in some sense to each other than to those in other groups (clusters). Then, you want to visualize the results in a plot. Is it right?

U: I don't know

B: From your request, I understood you want to apply to cluster, an analysis that aims at finding groups of data similar to each other (clusters). This kind of analysis doesn't require any additional information from you; it works with total autonomy.
Have I understood correctly your request?

U: no

B: I think I misinterpreted your original request. I will ask you some questions to better understand what you want to do. I see that you indicated the presence of a label in your dataset. Do you want to try to predict its value from the other data?

U: yes

B: Given the composition of your dataset, we will use a Classification Algorithm to predict the value contained in column class. Are you interested in the prediction itself, or are you more interested in understanding which are the most influencing factors in determining the prediction?

U: I want to see the influencing factors

B: Ok, we will perform a Feature Importance analysis to highlight which are the most important factors in the prediction outcome
I need one last piece of information: do you want to use all
the columns in your dataset or do you want to use only a subset
of them?

U: only a subset

B: Perfect! Do you want me to select the columns automatically?

U: no
B: Ok, I'll ask you about the columns later on. Let's proceed! List the features you want to remove using a comma to separate them:
gender, age, hypertension,heart_disease, ever_married,
work_type, Residence_type,avg_glucose_level, BMI,
smoking_status

U: gender, ever_married, work_type, Residence_type

B: Ok, I will consider only columns age, hypertension,
heart_disease, avg_glucose_level, BMI, smoking_status, class
B: The 33.366% of the rows have at least a missing value.
Do you want to REMOVE or FILL the rows with missing
values?

U: Remove

B: Ok, I will remove them. The 20.548% of the rows are
outliers. I will remove them

Table 3.4 – *Conversation between DSBot and the user after the interpretation of the first user's request failed (use case B).*

3.7.3 Use Case C: Troubleshooting

Alice, an ornithologist, was trying to perform a clustering analysis. As before, Alice navigates to the DSBot webpage, uploads her dataset, and fills in the necessary information; then, she writes her objective: "Split the data into groups." The system executes the following pipeline: `missingValuesRemove oneHotEncode outliersRemove laplace agglomerativeClustering pca2 scatterplot`.

At this point, the troubleshooting process starts, and she is presented with the results and the chat panel. The pipeline has identified two clusters, but the division line is not neat, and some samples are miscategorized: she writes: "*The division is not clear.*" The troubleshooting model uses the pipeline and the extracted problem type `wrong_cluster_division` to find the solution that suggests modifying the number of clusters or changing principal component analysis with another module and returns the sentence "*If you don't like how the clusters are positioned, try to change the number of clusters. If no other modifications satisfy you, try to change the principal component analysis algorithm. When you are ready, you can rerun the pipeline*".

Alice can now see the results and the chat panel as before, but also a representation of the pipeline, where the relevant parameters are highlighted. She edits the number of clusters and sets it to three. Then she reruns the pipeline. The page updates with the new results. Alice is now satisfied with the outcome and can continue her studies. Figure 3.11 shows the system interface.

3.8 Evaluation

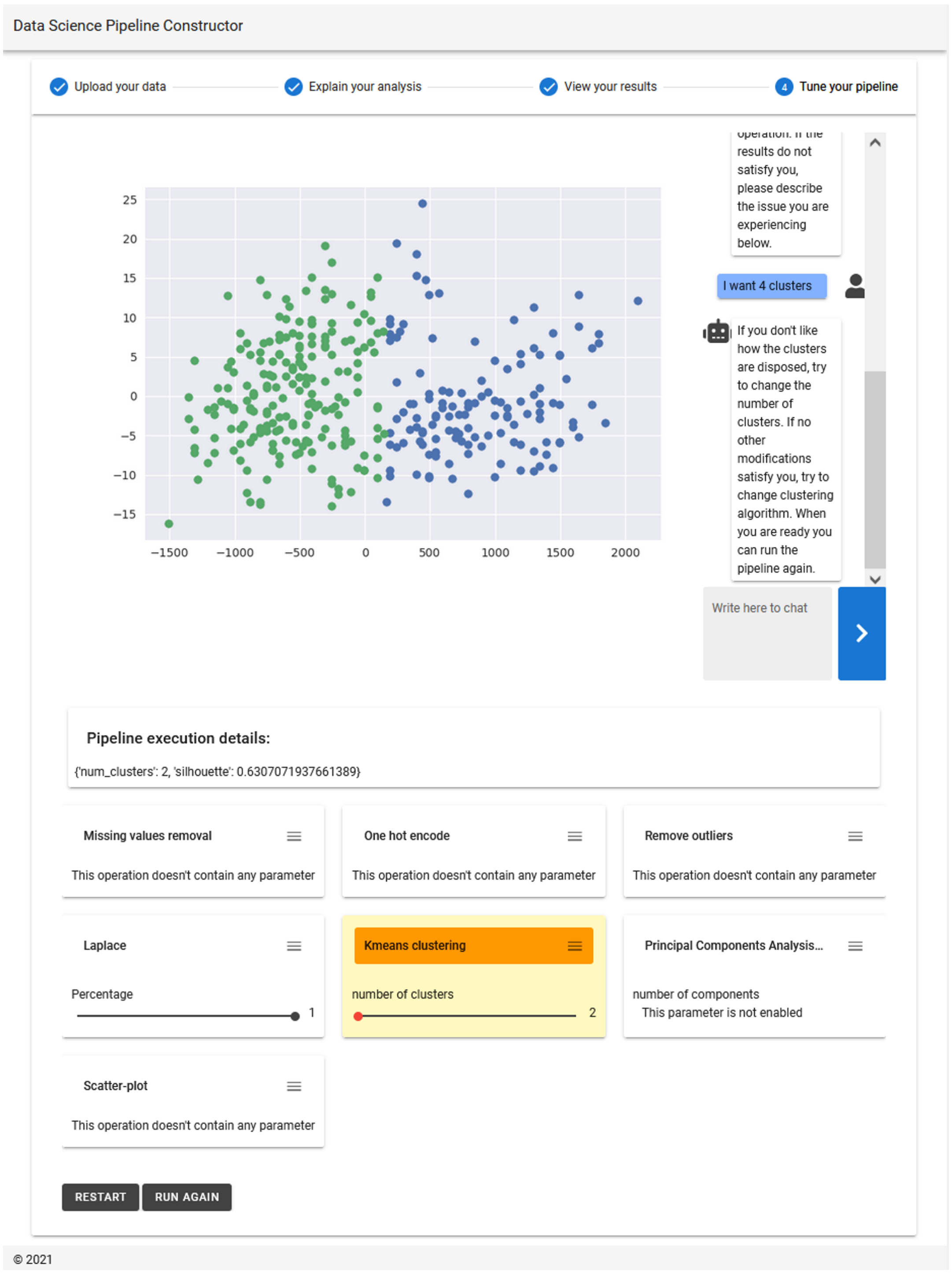
We conducted three evaluations with different objectives: (i) to assess the accuracy and computation time of DSBot, (ii) to test the system's comprehension of the user's research questions and its ability to convert them into precise executable DS pipelines, and (iii) to evaluate how the troubleshooting module was able to assist users who were not experts in data science in optimizing results.

3.8.1 Evaluation of the Automatic Machine Learning Pipeline Executor

We sought to validate DSBot's performance in terms of accuracy (for classification tasks) and RMSE with this investigation (for regression tasks). We also measured the execution time, which needs to be kept to a minimum to ensure a positive user experience.

We chose the well-known AutoML framework TPOT [149] as a starting point for our tests. It employs genetic programming to explore hundreds of ML pipelines intelligently and returns the one that optimizes a user-defined score function.

We studied a case in which the user allows DSBot to automatically choose one method and adjust its hyper-parameters while not specifying the algorithm for the classification or regression task.



operation. If the results do not satisfy you, please describe the issue you are experiencing below.

I want 4 clusters

If you don't like how the clusters are disposed, try to change the number of clusters. If no other modifications satisfy you, try to change clustering algorithm. When you are ready you can run the pipeline again.

Pipeline execution details:

{'num_clusters': 2, 'silhouette': 0.6307071937661389}

Missing values removal
This operation doesn't contain any parameter

One hot encode
This operation doesn't contain any parameter

Remove outliers
This operation doesn't contain any parameter

Laplace
Percentage 1

Kmeans clustering
number of clusters 2

Principal Components Analysis...
number of components
This parameter is not enabled

Scatter-plot
This operation doesn't contain any parameter

Figure 3.11 – *The layout of the results page. The user can interact with the chatbot on the right while viewing the analytic results on the left. The parameters associated with the user-identified problem are highlighted in the pipeline at the bottom.*

The evaluation was conducted using a combination of datasets from Kaggle[3] and those used in the evaluation of TPOT by its authors[4]. The final collection includes 12 datasets for regression and a total of 18 datasets for classification. Additionally, we choose the dataset collection to be as diverse as possible, encompassing a wide range of topics.

Note that TPOT only analyzes datasets without missing values or categorical features (which must be encoded in advance) to determine either the best classifier or the best regressor. In contrast, DSBot is an end-to-end tool capable of performing a complete analysis, including data cleaning, data preprocessing, and results visualization.

To enable the comparison, we gave the original datasets to DSBot; for TPOT, we used an *Iterative Imputer* approach to fill in the missing values and a *one-hot-encoder* to encode the categorical variables. Then, we ran the subsequent workflow 50 times for each dataset, averaging the outcomes:

1. We used the 20% of samples we randomly picked from the dataset as the testing dataset and the other samples as the training set;
2. We use five generations of populations of 50 pipelines to execute TPOT on the training set; a typical TPOT pipeline would comprise feature selection, feature engineering, model selection, and parameter adjustment.
3. We use the training set to run DSBot. It automatically creates a pipeline that goes from data preprocessing through data visualization; we utilized the `autoClassification` (`autoRegression`) module as a classifier and regressor. After choosing the model, we terminated the pipeline because we were not interested in the result presentation. A typical DSBot pipeline may comprise feature selection, encoding of categorical features, outlier elimination, and several methods to handle missing values (impute, delete).
4. We did not intervene during the execution, since the testing dataset were chosen for not having any of the characteristics that would have made necessary user's interaction in DSBot pipeline.
5. Both techniques produce a pipeline for a classifier or regressor. The time it took the two systems to create their candidate models was saved by us.
6. Using the held-out dataset, we used the two candidate models and assessed accuracy for classification tasks and Root Mean Squared Error (RMSE) for regression tests.

Tables 3.5 and 3.6 present aggregate results in terms of performances and execution times. Each entry in both tables represents a dataset on which the pipeline was run using both TPOT and DSBot. The dataset dimensions (rows × columns), the average performance relative to 20 runs with DSBot with its standard deviation (accuracy for classification, RMSE for regression), the performance comparison between DSBot and TPOT (DSBot mean performance - TPOT mean performance), the average time required to execute one

[3]https://www.kaggle.com

[4]http://www.randalolson.com/data/benchmarks/

Dataset	**rows × columns**	**Mean (std) accuracy**	**Δ mean acc.**	**time [*sec*]**	**% time TPOT**
Yeast	1479× 8	0.5939 (3.91e-02)	-0.0134	3.17	1.77
Vowel	990× 13	0.9803 (1.00e-02)	**0.0657**	2.83	1.08
Vehicle	846× 18	0.8055 (3.67e-02)	**0.1432**	3.98	3.49
Breast cancer	286× 9	0.7008(5.34e-02)	-0.0292	1.05	3.02
Diabetes types	768× 8	0.7662 (3.91e-02)	**0.0341**	2.08	4.46
Cleveland nom.	303× 7	0.5688 (5.48e-02)	**0.0155**	1.51	3.42
Balance scale	625× 4	0.8904 (2.45e-02)	-0.0172	2.51	4.72
Vote	435× 16	0.9557 (2.36e-02)	**0.0086**	1.86	4.56
Chess	3196× 36	0.9860 (5.67e-03)	**0.0011**	35.21	17.94
Stroke	5110× 11	0.9500 (8.08e-03)	**0.0023**	223.38	131.11
Australian	690× 14	0.8500 (4.40e-02)	**0.0094**	3.23	6.44
Ecoli	327× 7	0.8530 (4.04e-02)	-0.068	1.50	3.24
Car evaluation	1728× 21	0.9528 (2.18e-02)	-0.0179	4.48	2.31
DNA	3186× 180	0.9498 (8.09e-03)	**0.0091**	63.91	5.89
Diabetes	768× 8	0.7701 (2.79e-02)	**0.0406**	2.53	5.43
Dermatology	366× 34	0.9479 (2.42e-02)	-0.0169	2.02	2.79
Adult	48842× 14	0.8661 (2.20e-03)	-0.0042	669.24	53.14
Ann thyroid	7200× 21	0.9962 (1.42e-03)	**0.0099**	59.56	12.41

Table 3.5 – *Evaluation of classification tests.*

run with DSBot, and the execution time comparison between DSBot and TPOT (DSBot mean time/TPOT mean time ×100) are all contained in the columns.

Vowel, *car*, *diabetes kinds*, *cleveland nominal*, *vote*, *chess*, *stroke*, *australian*, *dna*, *dermatology*, and *ann thyroid* datasets were among the 11 out of 18 datasets for which DSBot performed higher in terms of accuracy. We achieved an accuracy of more than 95% in six of these cases. Only *cleveland nominal* achieved a poor accuracy (56 percent), while the other four earned an accuracy between 75 and 85 percent. When DSBot performed worse than TPOT, it always achieved comparable accuracy (within 95% of TPOT's accuracy).

Additionally, the execution time is faster than TPOT. On average, DSBot spent 14.85% less time performing classification tasks than TPOT for the same analysis, while it only used 3.46% as much time to perform regression tasks.

We got superior results with the regression on 8 out of 12 datasets. In these scenarios, DS-Bot outperforms TPOT by obtaining a reduced RMSE in a shorter amount of time.

3.8.2 Evaluation of the translation into executable pipeline

We developed a dataset of research questions in natural language, each accompanied by a description of the operations used to respond to those inquiries, to test the system's capacity to translate a human inquiry into a workflow of activities. We took advantage of the Kaggle[5] platform's dataset area, which allows users to upload datasets and Python notebooks that analyze them.

[5]https://www.kaggle.com

Dataset	**rows × columns**	**Mean (std) RMSE**	**Δ mean RMSE**	**Time [*sec*]**	**% Time TPOT**
Sample regression ds	10000× 21	1.40e-01 (8.92e-02)	4.51e-02	116.77	11.89
Students performance	1000× 8	2.69e-09 (5.99e-11)	**-1.71e-11**	1.62	3.31
House price	545× 12	1.11e+06 (7.32e+04)	**-1.14e+05**	1.56	3.43
Real estate	414× 7	8.09e+00 (1.81e+00)	4.82e-01	1.04	1.80
Material strength	1030× 8	5.54e+00 (8.96e-01)	3.06e-01	2.24	3.62
Patients LOS	835× 4	2.51e+02 (1.14e+02)	1.49e+01	2.35	2.54
Possum length	104× 13	2.08e+00 (2.29e-01)	**-7.79e-02**	0.53	1.41
Insurance price	1338× 6	4.60e+03 (3.99e+02)	**-8.1e+02**	2.10	2.50
Boston houses	506× 13	3.48e+00 (7.30e-01)	**-8.67e-02**	1.42	2.78
Startup marketing	50× 4	9.02e+03 (2.76e+03)	**-2.54e+02**	0.29	1.13
Insurance expenses	1338× 6	4.62e+03 (4.70e+02)	**-8.52e+01**	2.13	3.31
Second hand cars	1000× 11	8.90e+03 (4.40e+02)	**-1.22e+02**	2.01	3.85

Table 3.6 – *Evaluation of regression tests.*

We filtered the datasets saved in.csv format and ranked them in order of community vote totals to create our evaluation. The 40 datasets with the highest number of votes were chosen, and those that DSBot could not analyze, such as those with temporal data or in which the data is supplied in more than one table, were discarded. Then, for each dataset chosen, we looked through the top 30 uploaded notebooks in search of a written description of the actions taken by the notebook. We link each research topic to the DAW-written pipeline used in the notebook. All of the research questions that DSBot did not support were disregarded. Where the operation was supported but not the specific algorithm (e.g., when a notebook used a neural network for classification), we used the high-level term to describe the operation (e.g., classification) instead of the specific algorithm. Eventually, the resulting test set contained 106 research questions on 13 different datasets.

We employed the research question as a test set for the seq2seq translator. The accuracy of each translated question was then assessed as follows:

$$accuracy = \frac{n^\circ correct}{n^\circ words} * 100. \tag{3.1}$$

A 62.1% accuracy on average was the result.

The two leading causes of errors are *translation errors*, which occur when DSBot misinterprets the research question, and *user errors*, which occur when operations carried out by users do not correspond to the research question expressed or the research question is too ambiguous in relation to the intended pipeline.

In both situations, the follow-up interaction is crucial to error recovery. In fact, the discussion does not stop at letting users select the procedures they like; it also offers more explanations and real-world examples to help users better grasp the processes. With an average accuracy of 97.7%, the conversational agent interface was able to rectify a large number of incorrectly translated sentences.

Alternative approaches to translating the natural language inquiry were also tested. In fact, we assessed four Transformers offered by the OpenNMT library that is intended for translation:

- Eight heads of multi-head attention were used to build the OpenNMT transformer model (T1). It displayed a 43% accuracy rate;
- Eight heads of multi-head attention were used to build the OpenNMT transformer model (T2). It was accurate to within 33%;
- The smallest OpenNMT transformer model was constructed with four heads of attention, a dot attention type, and sixty-four smaller-sized feed-forward hidden layers (T3). It achieves a 29% accuracy rate;
- GPT-2 model that has been trained with 355 million parameters and fine-tuned using our training set. Its accuracy was 62.9 percent, an improvement over our method of 1.4 percent. Limiting the generated output to contain terms from the workflow language is impossible because GPT-2 is primarily a text generator. GPT-2 generates an incorrect pipeline in about 2 percent of cases, which would have been useless for our purposes.

In conclusion, the strategy outlined in DSBot is suitable for the demands of our research. Our comparison investigation revealed that: (i) three of the four examined transformers are less accurate than our LSTM model; (ii) the fourth transformer, GPT-2, uses a lot more resources than our method and produces less reliable pipes.

3.8.3 Evaluation of the Troubleshooting process

This assessment's purpose is to assess how well the support for troubleshooting works. We chose to target individuals with some scientific training but little to no expertise in the field of data science to test the suggested approach. In contrast to people with some or significant data science knowledge, referred to as *experts* in this section, we shall refer to them as *non-familiars*.

Research Questions Definition

This investigation focuses on three key areas: first, it is crucial to comprehend what non-familiars and their interactions with the DSBot system think about the findings. After that, we are curious about how individuals convey what they enjoy and do not like. Finally, concerning the suggested remedy, we want to evaluate how well the new system works at directing users toward better outcomes.

Users' focus. An experienced data scientist knows what to look for in an analysis result and can take the appropriate steps to address any problems. However, because it is intended for non-familiar people as well and aims to assist them in resolving problems on their own, it is crucial to recognize the aspects on which they concentrate because only in relation to these aspects will they naturally express their opinions. In contrast, they will require additional guidance to recognize other characteristics of the outcome.

Users' expression. Once it is evident which features are essential to outsiders, it is crucial to comprehend how they articulate themselves, including whether they speak of the problem they want to address or the solution they intend to use. Additionally, we want to evaluate whether this information varies according to the subject's background.

Interface Evaluation. The last objective is to test the proposed solution in its components: the conversational agent and the pipeline representation. We want to check whether the users can improve the results by changing parameters or modules in the pipeline, whether they feel that the chatbot supports them or is of hindrance, and finally, what they do if they are not given a specific task.

The following is a synopsis of the research questions:

1. How do non-familiars evaluate the results?
    (a) Do they understand the results of the analysis?
    (b) What is their focus?
    (c) Are there important aspects that are not considered?
2. How do non-familiars interact with the conversational assistant?
    (a) What type of sentences do they use?
    (b) Do they mention specific aspects of the problem or remain generic?
    (c) Does this change depending on the experience level of the subjects?
3. Is the proposed solution effective?
    (a) Can the subjects improve the result by changing the pipeline?
    (b) Is the conversational agent useful in solving the issues?
    (c) How do the subjects behave if not given a specific task?

Participants

We chose 12 participants for this study from various academic backgrounds, including computer science, design, and psychology, as well as from various occupations, including undergraduates, Ph.D. candidates, and post-docs, as shown in Table 2. While two individuals were familiar with statistical analysis, all the subjects claimed little to no knowledge of data science.

Participants who had no prior knowledge of data science would have had trouble understanding the examiner's requests, while testing experts would have produced false results regarding the usefulness of the problem-solving agent because they would not have needed it. This group of users represented the potential system users.

Four datasets were created before the experiment, each with a different interpretation of the data but equivalent in content: research on a graphical interface, a comparison of TV

Table 3.7 – *Background and occupation of the participants to the test*

4	Students with a background in computer science
4	Ph.D. students with a background in computer science
2	PhDs without a background in computer science
2	Research assistants without a computer science background

shows, data from a remote exam session, and a collection of books. A card was printed with information about each dataset and an illustration of what it included.

DSBot was slightly modified to generate the identical clustering analysis pipeline each time, which consists of a principal component analysis module and an agglomerative clustering module with a number of clusters parameter. Instead of the optimal value for this dataset, which is two, there are nine clusters instead. Furthermore, due to its subpar performance, the principal component analysis module creates a mixed-cluster representation. Multidimensional scaling is the most effective module to utilize. Sessions with users were organized as follow:

1. Initial Interview: Questions regarding the subject's educational background, present employment, and competence with data science and statistical analysis are asked.
2. Dataset selection: The four dataset cards are provided to the respondent, who is then asked to select the one they feel most at ease with after hearing about each dataset's specifics.
3. DSBot explanation: The user is given a brief overview of the DSBot tool, enabling data science analysis without needing specialized knowledge. The selected dataset is uploaded to the DSBot main page, and the predetermined research question, "Find the groups in the data," is added. The user can access the information from the initial dataset entries printed on paper.
4. First impression (Figure 3.12): following the pipeline execution, the user is prompted with a series of questions to assess what stands out in the outcome: What does the outcome seem to like to you? How about it? And what about this outcome would you wish to see improved?
5. First issue: It is noted that there are too many clusters if the subject is unaware of it.
6. First interaction (Figure 3.13): The user is asked to interact with the system to resolve the problem after it has been mentioned that the chat panel can assist in resolving issues with the results (while being careful not to influence his interactions with the chatbot). In reality, the subject is anticipated to ask the chatbot a question that will draw attention to the parameter for the number of clusters. After then, the individual is anticipated to edit the parameter.

7. Second opinion (Figure 3.14): The updated findings are displayed once the user has successfully decreased the number of clusters. The identical inquiries as in step four are posed to the topic.
8. Second issue: The fact that the clusters are intermingled is brought up if the subject is unaware of it.
9. Second interaction: To resolve this second problem, the user must engage with the system. Again, the subject is intended to use the chatbot to ask a question and switch the module from principal component analysis to multidimensional scaling using the interface.
10. Third opinion (Figure 3.15): The subject is quizzed once more on the step 4 questions following the execution of the modified pipeline.
11. Free task: The user is instructed to experiment with a new dataset (different from those used in the preceding steps) until they are happy with the outcomes, providing feedback on the activities taken and the outcomes received.
12. Assessment Questionnaire: The subject is requested to complete a questionnaire that includes a usability test (SUS [180]), a task intensity test (TLX [181, 182]), and open-ended questions regarding the experience.

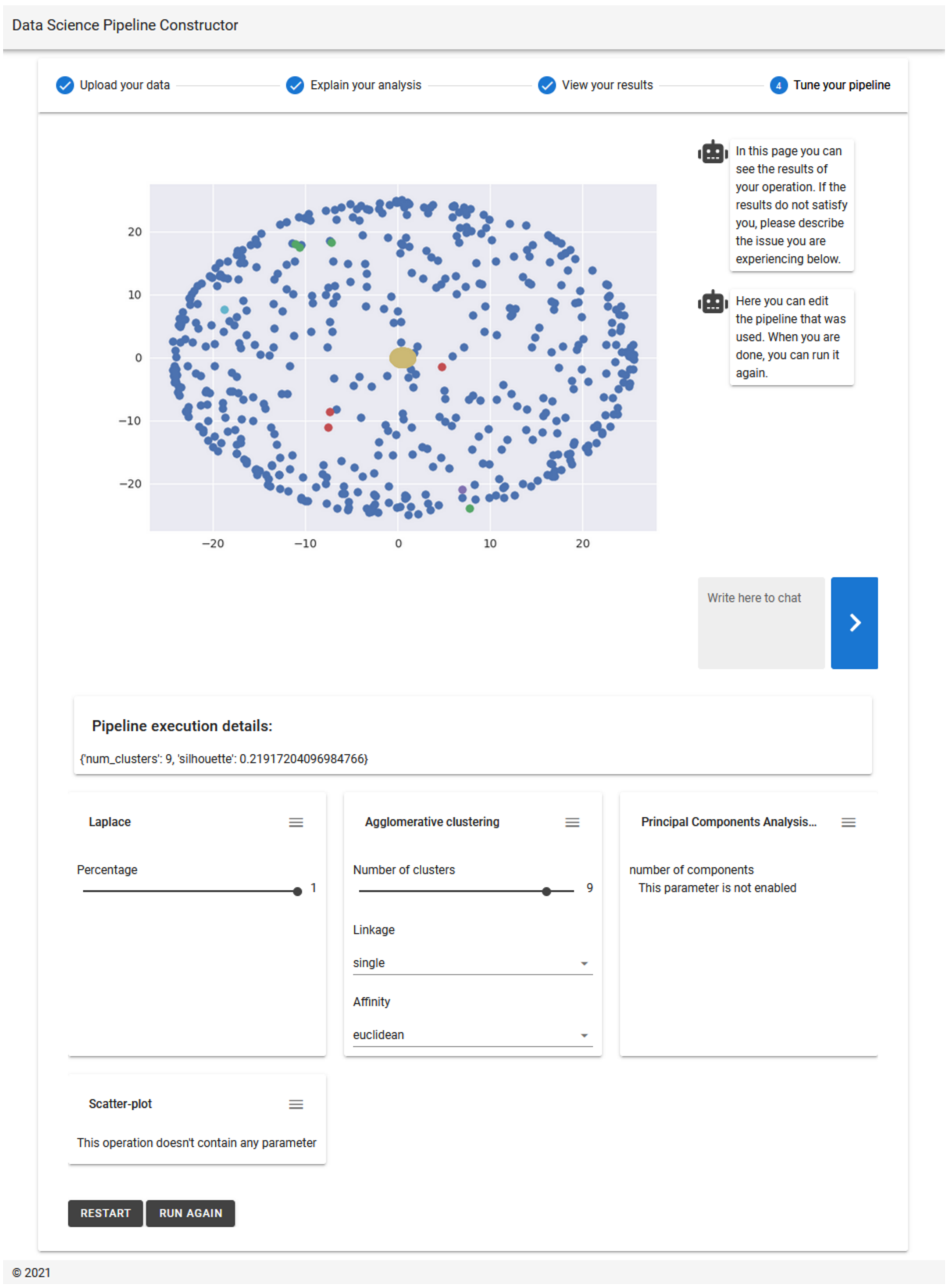
3.8.4 Results

The examination took an average of 28 minutes (with a minimum of 13m52s, a maximum of 43m41s, and a standard deviation of 6m55s, as detailed in Table 3.8). Here, the qualitative data—notes made during the test's execution, sentences used with the conversational agent, user comments, and their final opinions—and the quantitative data from the surveys are provided.

Quantitative Data

Experience Score. Each participant received a score depending on how well-versed they were in data science and statistical analysis after the preliminary interview. The results are used to determine which topics can be classified as unfamiliar: a combined score of four or more designates people with a non-negligible background in data science and statistics, while the other results are considered unfamiliar. The scores ranged from zero (completely unaware) to three (familiar). Of the 12 participants, seven were classified as unfamiliar, while the remaining five had more data science and statistics backgrounds.

SUS. Users received a System Usability Scale questionnaire [180]. It consists of 10 questions rated from 1 (completely disagree) to 5 (totally agree). The results were handled according to the standard procedure: the answers to positive questions had to be modified by deducting 1, while the answers to negative-worded questions had to be modified by deducting from 5. All of these results were then added up for each user and multiplied by 2.5 to obtain a score ranging from 0 to 100. The global SUS score, calculated by averaging these values, was 73.75, with a standard deviation of 13.29.



In this page you can see the results of your operation. If the results do not satisfy you, please describe the issue you are experiencing below.

Here you can edit the pipeline that was used. When you are done, you can run it again.

Pipeline execution details:

{'num_clusters': 9, 'silhouette': 0.21917204096984766}

Laplace

Percentage 1

Agglomerative clustering

Number of clusters 9

Linkage: single

Affinity: euclidean

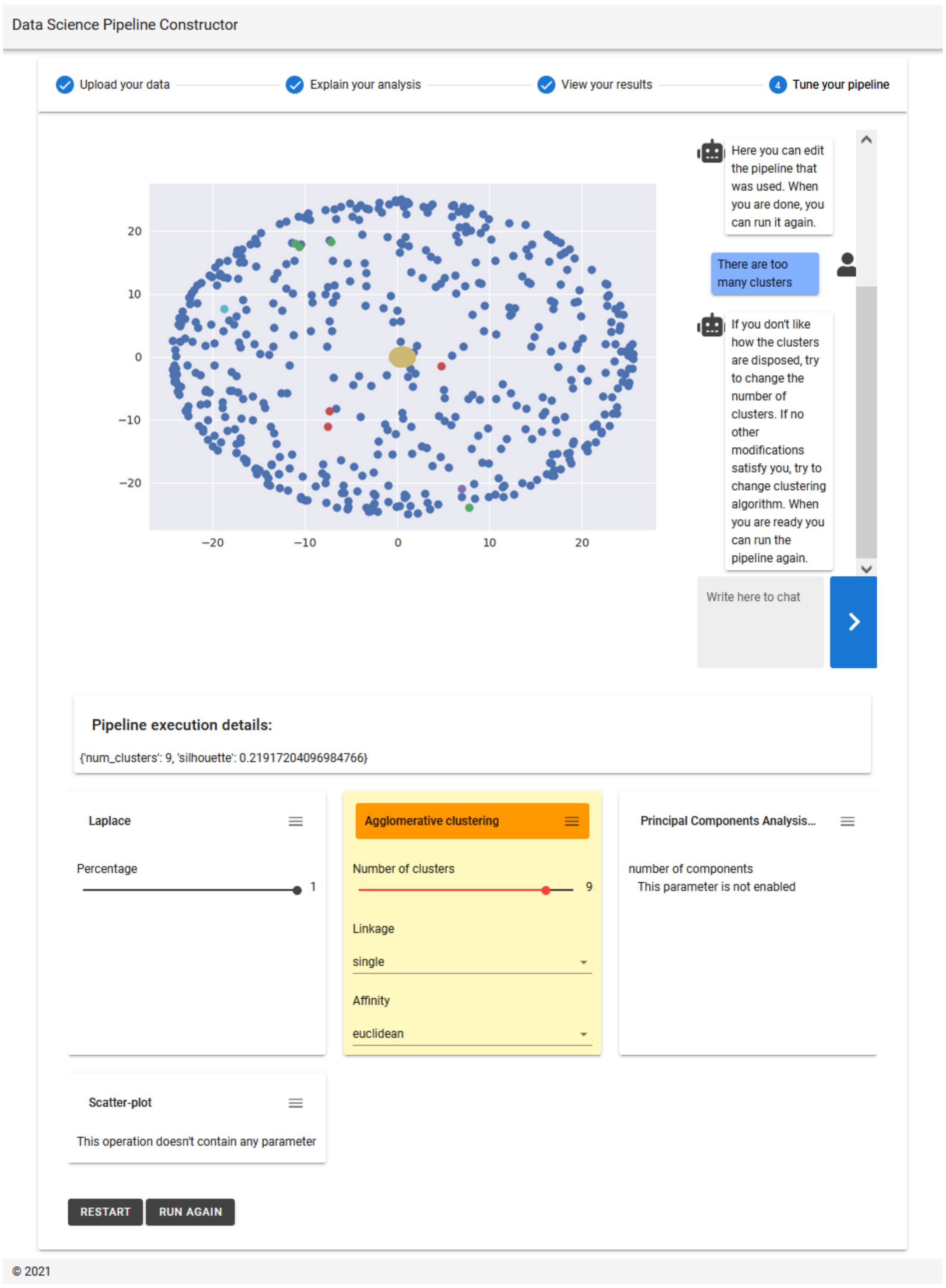
Principal Components Analysis...

number of components
This parameter is not enabled

Scatter-plot

This operation doesn't contain any parameter

FIGURE 3.12 – *Screen capture of the evaluation – initial results*


Data Science Pipeline Constructor

Upload your data — Explain your analysis — View your results — 4 Tune your pipeline

Here you can edit the pipeline that was used. When you are done, you can run it again.

There are too many clusters

If you don't like how the clusters are disposed, try to change the number of clusters. If no other modifications satisfy you, try to change clustering algorithm. When you are ready you can run the pipeline again.

Write here to chat

Pipeline execution details:

{'num_clusters': 9, 'silhouette': 0.21917204096984766}

Laplace

Percentage: 1

Agglomerative clustering

Number of clusters: 9

Linkage: single

Affinity: euclidean

Principal Components Analysis...

number of components

This parameter is not enabled

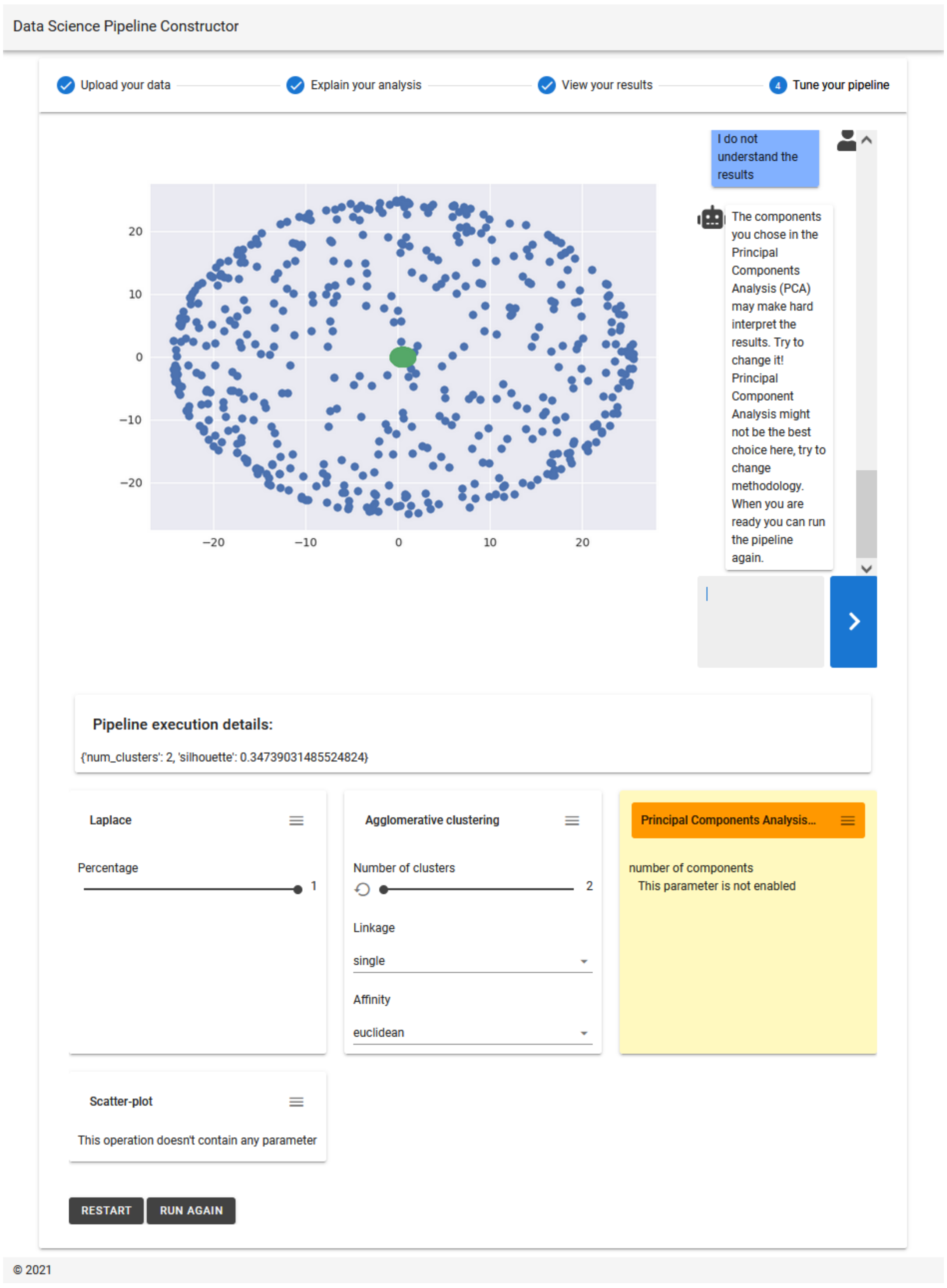
Scatter-plot

This operation doesn't contain any parameter

RESTART RUN AGAIN

© 2021

Figure 3.13 – *Screen capture of the evaluation – first interaction*



I do not understand the results

The components you chose in the Principal Components Analysis (PCA) may make hard interpret the results. Try to change it! Principal Component Analysis might not be the best choice here, try to change methodology. When you are ready you can run the pipeline again.

Pipeline execution details:

{'num_clusters': 2, 'silhouette': 0.34739031485524824}

Laplace

Percentage 1

Agglomerative clustering

Number of clusters 2

Linkage: single

Affinity: euclidean

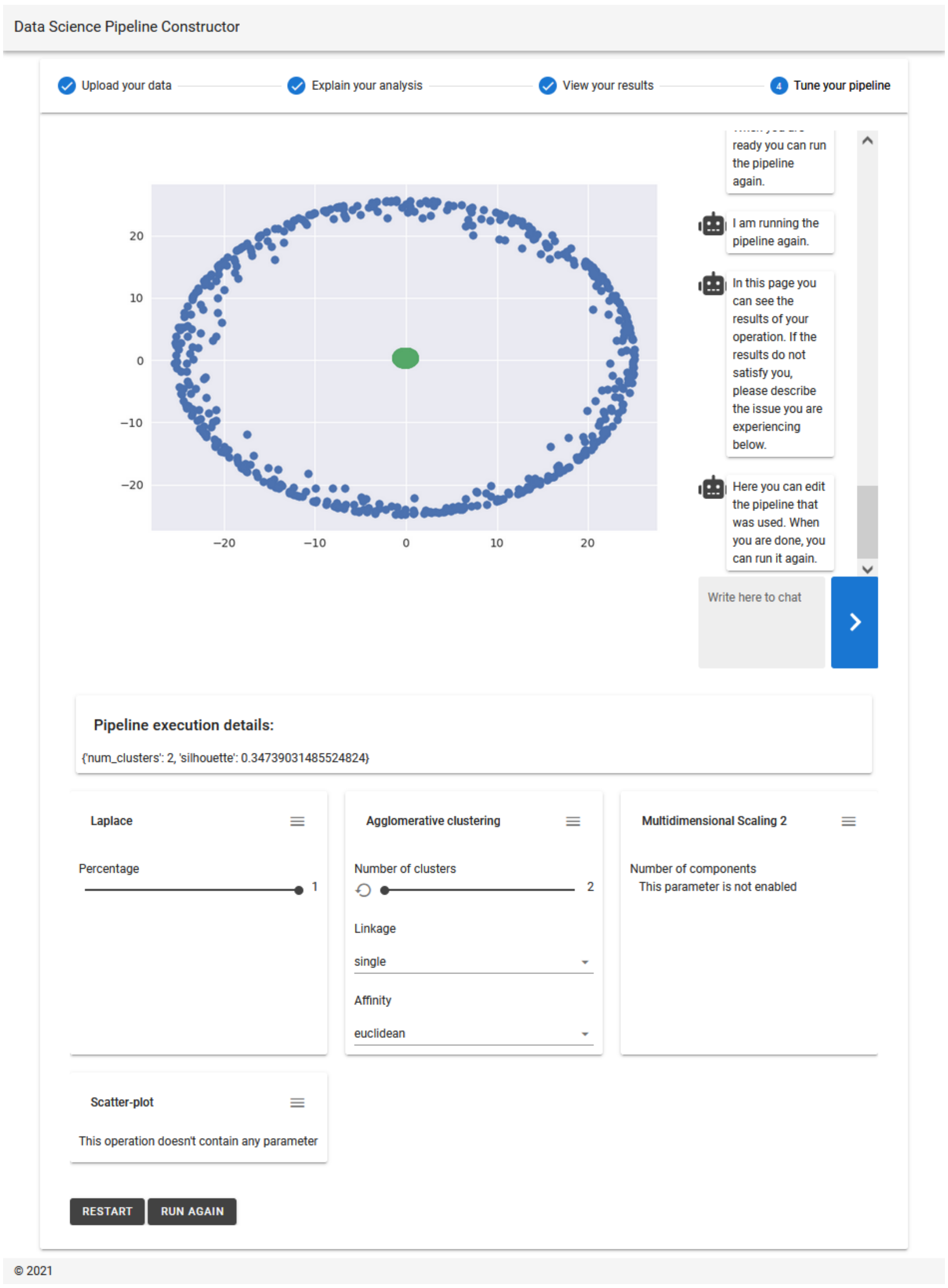
Principal Components Analysis...

number of components
This parameter is not enabled

Scatter-plot

This operation doesn't contain any parameter

FIGURE 3.14 – *Screen capture of the evaluation – second results*



ready you can run the pipeline again.

I am running the pipeline again.

In this page you can see the results of your operation. If the results do not satisfy you, please describe the issue you are experiencing below.

Here you can edit the pipeline that was used. When you are done, you can run it again.

Pipeline execution details:

{'num_clusters': 2, 'silhouette': 0.34739031485524824}

Laplace

Percentage: 1

Agglomerative clustering

Number of clusters: 2

Linkage: single

Affinity: euclidean

Multidimensional Scaling 2

Number of components

This parameter is not enabled

Scatter-plot

This operation doesn't contain any parameter

FIGURE 3.15 – *Screen capture of the evaluation – final results*

Table 3.8 – *Quantitative data collected during the evaluation*

	Duration	Experience	SUS	RTLX
Min	13:52	2	55	10
Max	43:41	5	95	61.67
Avg	28:02	3.25	73.75	34.86
σ	06:55	1.09	13.29	16.05

RTLX. A workload assessment was also conducted using the Raw Task Load Index [181, 182]. The test has six questions with scores ranging from 1 to 10 in its most basic form. Each user's responses were averaged, multiplied by 10, and given equal weights to measure their perceived workload. The final RTLX value, which is 34.89 with a standard deviation of 16.05., is then calculated by averaging the values. Table 3.8 lists the experiment duration's minimum, maximum, and average values and the users' experiences and survey responses.

Qualitative Data

The subjects' pertinent comments and some notes were recorded during the experiment. A total of 93 comments and notes, 34 user opinions, and 37 chatbot interactions were gathered and elaborated as detailed in the following paragraphs.

Chatbot Interaction. The sentences the subjects used to communicate with the conversational agent during each stage of the experiment were clustered in two groups, as *problem related* or *domain related*, depending on whether they described data science issues that the natural language understanding component would easily recognize (cluster, division, distribution, dimensions), such as *"I don't like the groups"*, or they were issues related the interpretation of the results, such as *"Why are horror and love movies together?"*. When operated on large corpus of data, this practice is called *Intent Mining* [183, 184].

Table 3.9 displays the outcomes. Additionally, according to their intended meaning, sentences were categorized into five groups, which are here provided with an example and an explanation of the logic:

- Inquire, "Can you change how data are shown?": the user asks what activities are possible and the system's capabilities are.
- Understand, "I don't understand what the dots are": the user acknowledges their confusion and requests a further explanation.
- Dislike, "I do not like what I am seeing": the user criticizes something.
- Want, "I expected more clusters": the user expresses what is desired or anticipated.
- Assert, "Clusters are not well separated": the user claims that the results are incorrect.

Table 3.9 – *Chatbot interaction sentences analysis*

	Inquire	Understand	Dislike	Want	Assert	DS	Domain
Count	2	6	8	9	12	21	6
Exp Avg	4.00	3.17	3.75	3.33	3.34	3.48	3.50
Exp σ	0.00	1.34	1.56	1.33	1.32	1.43	1.26

This table reports the number of sentences for each of the intents and tags, the average experience score associated with the subject who used the sentences, and its standard deviation.

Table 3.10 – *Focus of users' sentences*

Focus	Count	Exp Min	Exp Max	Exp Avg	Exp σ
Cluster separation	37	2	6	3.81	1.41
Image	24	2	6	3.13	1.24
Axis	13	2	6	2.92	1.14
Cluster size	12	2	6	2.83	1.14
Cluster number	12	2	6	3.42	1.38
Unassigned	4	2	3	2.25	0.43

Focus. The lines used with the chatbot and the comments were tagged with their focus, as shown in Table 3.10, to understand better which aspects are important to the users. When an utterance addressed two different things, it was counted twice—once for each category. Six different groups were found:

1. Cluster separation: the clusters' arrangement is discussed in the sentence.
2. Image: the phrase refers to the outcomes as a whole without going into any specifics.
3. Axis: the sentence mentions the axis of the results, the dimensions, and their meaning.
4. Cluster size: the results' axes, dimensions, and significance are mentioned in the clause.
5. Cluster number: the number of clusters mentioned in the sentence.
6. Unassigned: the sentence refers explicitly to the largest cluster, which three users perceived as being unassigned and not a group member.

Four comments exhibited wonder; others expressed perplexity regarding the results (7) or the interface, and the remainder remarks did not fit into any emphasis (5). Two comments gave respect to unimportant features like the color scheme and the geometric beauty of the outcomes.

Table 3.11 – *User opinions on DSBot by type*

Tag	Count	Description
+ Intuitive	2	The interface is intuitive and easy to understand.
+ Highlight	3	The highlighting mechanism is useful and well-integrated
+ Other	4	The editable pipeline representation is well-done
– Issues	2	There are issues with the interface
+ Valuable	6	The chatbot adds value and is helpful
– Expert	3	The chatbot is not enough; there is still the need for an expert
– Variation	2	The chatbot answers need more variation

This table shows the results of the analysis of the user opinions. The tag is preceded by a + if it is a positive comment and a – if the opinion is negative.

Notes. According to the notes made during the experiment, the subjects' understanding of the meaning of the results was unclear on three separate occasions: two did not comprehend that different colors indicated different clusters, and one kept confusing the number of clusters with the dataset's dimensions. Four individuals freely edited the relevant pipeline pieces after understanding on their own that the chatbot was doing so; these subjects also displayed confidence in the free task. Two of the twelve subjects needed help finishing the second exercise, particularly identifying the issue with the results.

Opinions. The subjects' opinions were divided and numbered by topic after the experiment, and the findings are shown in Table 3.11 opinion. The subjects were asked to identify the system's positive and negative elements. When a user made an identical point more than once, the opinion was only counted once for each issue. It is significant to note that the results in this section cannot be put together. One could assume that respondents who did not express a good opinion on a certain feature did so because they held that opinion to be unfavorable, but this was not the case.

On the other hand, 2/12 users had issues with specific aspects of the interface not directly related to the functionalities of the system: for instance, the fact that the pipeline was not visible without scrimping; 9/12 subjects liked the representation of the pipeline, and the fact that it was editable; of these, 3 also explicitly mentioned as a positive aspect that the relevant elements of the pipeline were highlighted; and 2 praised its intuitiveness. Regarding the conversational agent, 6/12 participants believed it provided value to the system and were excited that it allowed for data science analysis. However, 3/12 said they still needed expert guidance, and 2/12 asked for more variety in the chatbot's responses. The fact that 2/12 stated they had trouble articulating what was wrong with the data is significant. Four out of twelve individuals asked for new features, including more flexible conversational agent engagement and more information on the outcome.

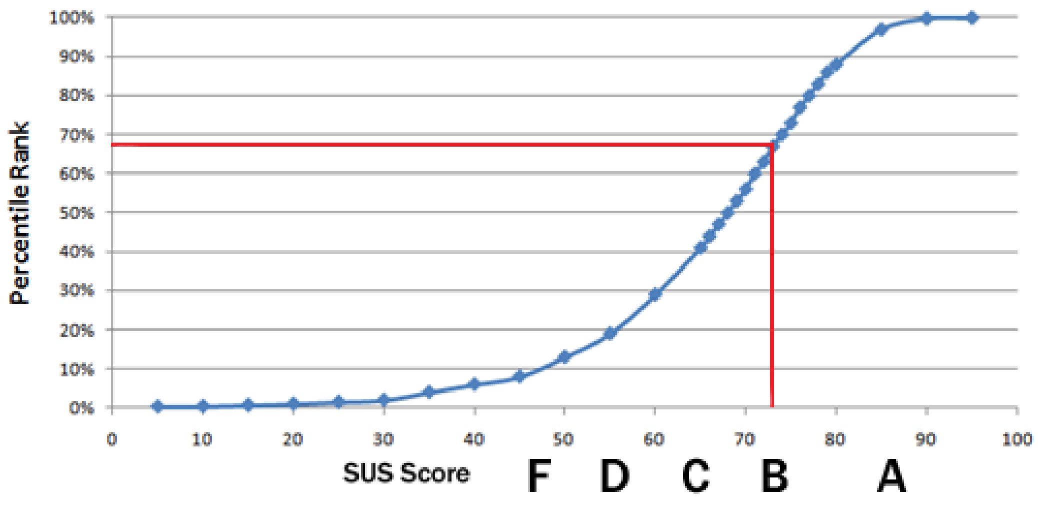


Figure 3.16 – *SUS percentile evaluation, adapted from [185]*

3.8.5 Discussion

Looking more closely at the data, it can be seen that the lowest score was related to needing the support of an expert, while the highest score was related to the integration of the various components, with the SUS questionnaire yielding a result of 73.75, which corresponds to the 68th percentile, giving a final mark of B(-) [185]; as reported in Figure 3.16. This is consistent with the findings from the analysis of user opinions: the interactive representation of the pipeline is highly valued, particularly the way that it highlights the best options based on the chatbot conversation; however, it is insufficient for some subjects, who require additional assistance in identifying and resolving the problems.

The RTLX evaluation scored a value of 34.89. Compared to the research by Grier et al. [186], this score falls comfortably within in the first quartile for cognitive tasks and the second one for computer activities, as shown in Table 3.12. This indicates that the activity is not excessively cognitive-demanding. DSBot and the additional functions were successful in making the data-science process accessible to non-experts.

We wanted to see whether the level of confidence with chatbots had an influence on the experience with DSBot. Dividing users in two groups according to the familiarity with chatbots they declared, no relation was found in an unpaired t-test that compared the individuals' self-perceived level of expertise with chatbot (clustered in *low* and *high*) with the SUS mark (p = .93), and the same is true for the RTLX score (p = .47). We can conclude that this experiment did not reveal any relationship between a user's experience and the task's perceived usability and difficulty. For this reason, we cannot draw conclusions on which target population our system is more effective. However, this study only looked at a small group of people with comparable levels of expertise. Therefore further research is required to determine the full impact of experience in this area.

Table 3.12 – *Cumulative frequency distributions of TLX scores*

Description	Min	25%	50%	75%	Max
Cognitive Tasks	13.08	38.00	46.00	54.66	64.90
Computer Activities	7.46	20.99	54.00	60.00	78.00

This table, extracted from the work of [186], shows the cumulative frequency distributions of TLX scores for two relevant task types.

When the types of utterances used to communicate with the conversational agent were analyzed, it was discovered that most of them were of the *Assert* type, in which users state what they believe to be incorrect. These are similar to the *Dislike* group, which came in third, but the user, in this case, is frequently less specific when describing the problem. The complementary technique is to make a direct reference to the outcome desired, as is the case in the *Want* sentences, which are the second most common, and these two sets indicate one way of interacting with the system in which the focus is on the current situation. The fact that many subjects used precise terminology to identify the problem helped the conversational agent respond better to these utterances.

It is interesting to notice that participants attempted to explain the outcomes 6 times (6 attempts total; it also indicates that this behavior is slightly more frequent among non-familiars, as predicted) and attempted to describe the chatbot's capabilities twice. This implies—also taking into account user feedback—that developing a brief introductory presentation of the findings and the available alternatives could be required when the system is initially utilized.

The most prevalent component that the respondents were interested in is *Cluster Separation*. However, users were able to identify problems with cluster size and quantity. Unfamiliar persons tend to be more perplexed by the diagram's *axis* and the supposed unassigned dots. This is also true for the *Cluster Size*, maybe because more seasoned users did not consider this feature problematic.

The notes and user feedback analysis demonstrate that, generally, the system was successful in assisting unfamiliar users in carrying out a data science analysis; furthermore, in two cases, the subjects displayed a learning effect, picking up the solution from the chatbot for the first time and applying it on their own during the free task.

This, along with the fact that the dimensions of the result created confusion and that the second most common group of sentences is those without a specific focus, tells us that it is again necessary to introduce the analysis results better. However, it also shows that a more guiding approach, such as a troubleshooting procedure, would be potentially beneficial to introduce the analysis's findings better.

3.9 Conclusions

This work introduces DSBot, a unique method, and system for creating and running data analysis pipelines starting from user-uploaded datasets and natural language queries. The most important features of DSBot are its definition of DAW, a domain-specific language for describing data analysis pipelines; a machine translation based on a neural network to create a DAW sequence from the user's query; a matching algorithm to select the best matching pipeline from a dictionary of pipelines; a conversational agent interacting with the user as needed; and a multimodal conversational agent for troubleshooting.

In addition to performing the operations asked explicitly by the user, DSBot automatically enhances the pipeline to improve the outcome in a way understandable to a user who is not a highly skilled data analyst. By analyzing the evolution of this tool in two case studies—one to examine the benefits of discussion and the other in which DSBot chooses the best pipeline to address the user's research question—we present empirical proof of the tool's potential.

By allowing non-experts to use data science, DSBot paves the way for a new family of tools that will make data science more approachable and valuable to a broader audience.

3.9.1 Limitations and Further works

However, there are still some things that still need to be solved. In the current system, an unbalanced dataset results in no actions being made. Either implementing rebalancing techniques can solve this issue during the data preprocessing processes (such as downsampling or oversampling) or by computing various evaluation criteria (e.g., Matthew Correlation Coefficient). The automatic selection of machine learning models, which in the current system version is based on the Accuracy metric, is another area that could use improvement. In these situations, more informative metrics should be used in place of the Accuracy metric, which is inappropriate when the data sets are unbalanced (e.g., balanced accuracy). The DSBot is modular, so these extensions will be simple to create and implement in the future.

Other improvements focus on two main issues: i) expanding and improving the set of pipelines supported by DSBot and the operations supported by the system that is currently permitted on single-table data only, and ii) enhancing the conversational power of the chatbot in some ways, including to elicit a more significant number of user research questions, to sustain a more natural interaction, to increase the transparency of the AutoML processes, and to explain the system more clearly [187].

We intend to address the first issue by incorporating and automating more algorithms and analysis modules, such as those used for time series analysis or survival analysis, as well as by testing them on a variety of datasets to increase the number of supported pipelines, improve their quality, and offer the user more sophisticated computational support.

By gathering many fresh real-world research questions with different formulations and exemplary conversation flows, we will expand the training corpora for research question interpretation and elicitation for the chatbot. To find the ones better suitable for the AutoML approach of DSBot, we will explore the explainable AI techniques already in use [187]. They will assist in the creation of new chatbot conversational patterns [188] so that the user can receive explanations that make the analysis procedures and their results more transparent, understandable, and reliable.

Also the empirical evaluation with users presents some limitations. In fact, the results, despite promising, lack of a comparison with a baseline, computed from the interaction of a control group with traditional data science applications. In addition, the study focused on a fictitious problem, a new study with real-world problems and research questions is necessary to assess the validity of the application.

Yet, from the interaction perspective, the work in GeCoAgent and DSbot shows the potentialities of multimodality in conversational interaction. Hence, there is a necessity to structure the process of creating multimodal interfaces, as we will investigate in the following chapters.

Part II

Multimodal Conversational Agent: a Design Perspective

Chapter 4
Problem Space and Research Questions Definition

4.1 Lessons learned with the past experiments

So far, we have described the implementation and evaluation of two multimodal conversational interfaces for data science: GeCoAgent, a tool for extracting from databases and analyzing genomic data, and DSBot, to analyze open-domain data in tabular form.

The two interfaces brought several scientific contributions. With GeCoAgent, we showed how to use user interviews and ontologies to design conversational interfaces, how to use grammar to model and implement the interaction, and empirically shows the advantages conversational agents bring to bioinformatics, especially but not only to non-experts.

With DSBot, instead, we showed how to combine natural language processing, conversational interfaces, and Automatic Machine Learning to translate users' research questions in executable data science pipelines that provide results comparable with well-established AutoML libraries. In addition, we implemented and tested a multimodal conversational troubleshooting module that can assist users in improving their outcomes.

However, despite the two systems present profound differences, summarized in Table 4.1. The GeCoAgent's *domain* is restricted to bioinformatics tertiary analysis; DSBot, instead, is domain independent. Consequently, GeCoAgent knows the data it operates on, therefore being able to dialogue on them. At the same time, DSBot is agnostic on data, offering conversational interaction only on the operations users want to perform.

	GeCoAgent	**DSBot**
Domain	Restricted (Bioinformatics)	Open (Tabular data)
Chatbot Role	Actor	Guidance
Multimodality	Co-exist	Alternate
Chatbot Aim	Composing pipelines	Guiding users
Prior Knowledge	Domain + Data science	Domain
Conversation Structure	Grammar based	Dynamic trees

Table 4.1 – *Differences between the conversational agent of GeCoAgent and DSBot.*

We designed the two conversations with different *roles* inside the interaction with the system. In GeCoAgent, the chatbot is the actor, and the task progression is entirely based on the conversation, as users can manipulate data and perform operations only through the chat. Visual interaction supports the execution of the tasks and feedback once the operations are completed. DSBot, instead, the chatbot is the guidance. The task progresses through actions both on the conversational and the graphical interface. The conversational agent intervenes only in those phases in which users need detailed explanations or guidance in their actions, such as the choice of the data science algorithm and the optimization of the results. As a result, this application alternates moments of visual, conversational, and mixed interaction (visual+conversational) according to the specific phase of the interaction.

For this reason, the *multimodality structure* is also different. The conversational agent in GeCoAgent co-exists with the graphical user interface, i.e., it is always displayed. All the visual interactions are functional to the chat (e.g., users can click on the suggestions to copy those terms in the message they are typing). At the same time, DSBot's chatbot alternates with the GUI: it is active only in the phases users can interact conversationally.

Also, the *aim* of the conversation changes accordingly. In GeCoAgent, the conversations' goal is to let users compose their pipeline, translating natural language in execution blocks and combining operations as they prefer. In DSBot, the conversation assists users, guiding them in the pipeline that the system chose to answer the given research question and asking for data-dependent decisions.

Consequently, the *prior knowledge* required for users to use the applications is different. To use GeCoAgent, users need a complete understanding of genomic data and be able to select the suitable algorithm to answer their research questions. In DSBot, users only need an understanding of the data they are using. DSBot automatically selects the most suitable algorithm and asks users only to make those decisions that are strictly dependent on the nature of the data it operates on, like eliminating samples with missing values or filling them with arbitrary values.

The *conversation structure* changes as well. GeCoAgent is entirely based on a hierarchical grammar structure that allows users to compose operations as they wish, always following the constraints given by the grammar rules. DSBot follows rigid trees composed dynamically by the conversational engine according to the module inserted into the pipelines. Contrary to what happens in GeCoAgent, where the conversation is conducted by the user, here the conversation flow is determined by the block chosen for the pipeline and the characteristics of the uploaded dataset.

This comparison shows a considerable difference between the two conversational agents, although both can be described as "multimodal chatbots for data science." To the best of our knowledge, there is no design reference model to describe the design of conversational agents, especially in a multimodal setting.

For this reason, we believe it is necessary to introduce a reference model and a codified terminology for describing these applications to highlight the differences and similarities between the systems. Such a model, in addition to allowing comparison and description of existing systems, could play a vital role in designing new interfaces, acting as a communication tool between designers (UX and conversational) and developers.

The second part of my doctoral work focuses on this purpose: I want to explore the implications of multimodality and define a model that can be used to design and develop task-oriented multimodal conversational interfaces.

To do that, I will proceed in three steps. I will start by exploring the concept of multimodality and its implications on the interaction. I will continue by defining a model to describe multimodal conversational interfaces and produce an authoring tool to create such interfaces from their model-based representation.

The remainder of this chapter is structured as follows. First, I will describe the problem space, defining the key terminologies on which the work is based. It will continue by describing the multimodal continuum, a tool we have designed to describe multimodal conversational agents. This chapter will conclude by formulating the research questions that guide this part of the work.

4.2 Defining the Problem Space

4.2.1 Model-based Conversational Agents

We define model-based conversational agents as those agents whose conversation is structured on a set of rules [189]. This approach opposes generative conversational agents, whose conversation is automatically creative through generative methods that learn from a large corpus of interaction examples [190].

The model on which the conversation is based can be of various types, but all are united by the fact that they are hand-crafted. Many model-based conversational agents are based on ontologies [189]. Others, instead, are based on process representation, which we will introduce in Section 4.2.3.

4.2.2 Task-oriented Conversational Agents

Both GeCoAgent and DSBot are task-oriented conversational agents, as are the conversational agents we will consider in the rest of the work. Contrarily to *general purpose* conversational agents, whose primary purpose is to entertain users [191], task-oriented conversational agents aim at guiding users to accomplish a goal in one or more domain [192].

Since the research panorama has yet to converge on a unified vocabulary to describe conversational applications [193], we introduce the terminology we will use in the following chapters referring to task-oriented conversational agents.

Goal. We define the goal, the final reason why the conversational agent was designed, and, consequently, why the interaction takes place. For example, a chatbot on the website of an airline company might have the goal of guiding users to buy flight tickets, whereas the conversational agent on the web application of a bank might have the goal of helping users in making money transfers.

Task. We define tasks as the sub-goals a conversational agent is programmed for. For example, the conversational agent on the airplane company might support the following tasks: look for available flights, purchase tickets, and modify bookings.

Process. We define the process as the sequence of operations the user must accomplish to fulfill the task. The process can be explicitly specified in the specifications of the conversational agent or implicitly result from the constraints the users have to accomplish the tasks. In the airplane website example, the process necessary to purchase a ticket might be to choose the departure and arrival date and airport, select one of the flights, insert passengers' generalities (name, surname, etc.), then insert debit card information to make the payment.

A task-oriented conversational agent has one goal that might be declined on multiple tasks linked by a process. Also, the process is unique, even if the sequence of the operations to reach the goal may vary. It is the process itself that rules how the sequence may vary.

4.2.3 Process-driven conversational agents

We define process-driven conversational agents as the model-driven task-oriented conversational agent whose process is explicitly described in the design of the agent itself. Often they are also referred to as rule-based conversational agents [194]. The model specification introduces a precedence constraint on the various interactions and the operations the user can start in the conversation.

There are many possible model specifications. Most of them are graph-based, such as tree-like structures [55] and grammars [11]. Other specifications try to include in the representation the information that rules the various branches in the conversation. Many applications, such as BPMN [19, 195] exploit already existing formalization.

Process-based conversational agents are opposed to generative agents, where the interaction rules are not hand-crafted by the designer but are learned automatically by the system, trained on a base of interaction examples [196].

4.2.4 Multimodal interaction

Multimodal interaction refers to the use of multiple sensory modalities, such as speech, gesture, touch, and vision, in human-computer communication [197]. This type of interaction allows users to engage with computer systems using a variety of modalities, thereby enhancing the user experience and making the interaction more natural and intuitive [171]. Multimodal interfaces can be found in a variety of applications, including virtual assistants, video games, and mobile devices [198].

Multimodal interaction allows users to communicate with computers using multiple channels of input and output, which can enhance the richness and effectiveness of the interaction. For example, a user may use a combination of speech and gesture to perform a task, such as selecting an item from a menu or zooming in on a map [125]. By combining different modalities, multimodal interfaces can provide more robust and flexible interaction, accommodating different user preferences and contexts [199]. Additionally, multimodal interaction can help to overcome the limitations of individual modalities, such as speech recognition errors, by providing redundancy and fallback options [200].

4.2.5 The Multimodality Continuum

We aim at strongly integrating Conversational Interfaces with other modalities, starting from a well-defined process, where the modalities cooperate to support the user in the tasks. An interface is multimodal when it employs more communication channels to interact with the user [199]. These communication channels are said modalities [201] and can exploit the same or different medium.

Multimodal interfaces have been widely investigated in human-computer interaction because, since human's perception of the world is intrinsically multimodal, they are considered the most natural interaction paradigm [202].

For users, multimodal interaction is highly convenient since it allows them to alternate the modalities to adapt at best to users' task [203].

Past research works investigated many types of multimodal interactions, such as speech-gestures, touch-speech [204] multi-sensory environments [205, 206], virtual reality, and tangible interaction [207]. In this thesis, we concentrate mainly on combining a chat with a graphical user interface. However, most of the results we will discuss in the following chapters are generalizable to any combination of modalities that comprises a conversational interface (text or speech-based) and another modality.

To better understand the implications of the integration, we want to introduce the Multimodality Continuum to describe how Conversational Interfaces can relate to other modalities in interactive applications. To exemplify the model, we describe the interplay between a chatbot and a Graphical User Interface, but the Multimodality Continuum also applies to other modalities.

The Multimodality Continuum categorizes chatbots into different interaction paradigms according to the communication channels among the different actors in the system, as shown in Figure 4.1. This model maps these interaction paradigms in a uni-dimensional space, whose ends are the uni-modal interaction and the complete integration.

In uni-modal agents (Figure 4.1*(a)*), the user has a one-to-one dialogue with the chatbot, usually in a messaging system or in a dedicated interface.

Multimodal interfaces allow people to use multiple types of inputs at once to carry out a task, with the advantage that one input's weakness is overcome by the strength of another [208]. Many modalities in the same application allow users to perform faster and more precise queries, less prone to errors [125]. Multimodality increases the usability and performance of the application [209], letting users achieve complex tasks. Necessary condition for multimodality is the *interoperability* of the various communications channels [210], but it not sufficient: the design of the user experience must orchestrate the different loci of interaction.

When the chatbot is coupled with a GUI, the two interaction loci can exploit the communication channels differently. In the simplest case, all the interaction happens through the chatbot, which can only change the state interface, as represented in Figure 4.1*(b)*. An example is in [211], where the user can only modify the content on the visual interface through the mediation of the conversational agent. This pattern has a limited adoption since it results in lower usability: users are required to shift their approach from *direct manipulation* of the interface to *indirect management* - from operating on the interface telling the chatbot what to do on it - that is more time and effort consuming [212].

Sometimes the user interacts with the graphical interface and the chatbot, but the two modalities do not communicate and are independent (Figure 4.1*(c)*). They are two wholly separated entities. If the user applies some changes in the interface, the chatbot cannot see them, and, on the contrary, if the chatbot converses with the user, the interface cannot know what they are talking about. Many user assistance chatbots fall under this category: when users switch from the web page to the chatbot, the latter is not aware of what was happening on the GUI, and therefore users have to describe the issues they are requesting assistance for.

An enhanced version of the previous paradigms enables the chatbot to modify the graphical interface state through a direct communication channel, as shown in Figure 4.1*(d)*. When the conversation proceeds, the GUI is updated to provide consistent information to the user. This approach is widely used when the task outputs are data visualizations or multimedia content. The communication between the modalities is unilateral: if the user manipulates the visual interface modifying its state, the chatbot is not informed. Ava [56] exemplifies this paradigm: the application is a multimodal interface, but while the chatbot can communicate with the GUI, the GUI cannot communicate with the chatbot. Consequently, if the user performs any operation on the GUI, she cannot use the chatbot any longer since it is not aware of the new state of the system.

Our approach synthesizes the three multimodal patterns elicited above in the one described in Figure 4.1*(e)*: every actor can interact with each other. This means that, in the case of integrating a conversational agent and a GUI, the user can chat and act on the interface. The chatbot's state is synchronous with the interface's state; the chatbot can see what happens in the graphical interface and dialogue with the user accordingly. The interface can be modified directly by the user or according to the conversation between the user and the agent. The chatbot cooperates with the other interface channels because it does not lose the actions that the user performs through the interface, but it can continue the dialogue by asking for the correct thing at the right moment. The user can switch between the modalities as she prefers. For example, she can decide to use the interface and ask for

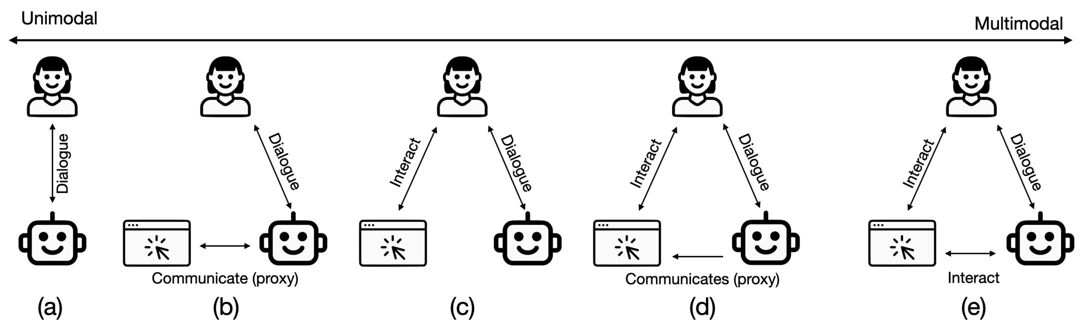


Figure 4.1 – *The Multimodality Continuum: from uni-modal to multi-modal fully integrated conversational interfaces. (a) Uni-modal conversational agent dialogues with the user. (b) The user either dialogues with the conversational agent or interacts manually with the interface. Both interactions are not contemplated in this model. (c) The three agents are linked: the user can either interact with the interface or dialogue with the agent, and the agent can send instructions to the interface. (d) One multimodal example of the conversational agent is the user dialogues with the agent, and the agent uses the interface as visual support. (e) All the possible connections are present: the user can either interact with the interface or dialogue with the agent, the agent can send instructions to the interface, and it can also receive information from the visual interface. Every agent can listen and interact with the other agents.*

help from the chatbot, or she can dialog all the time with the conversational agent and, at some point, change the input channel.

When multimodality is not complete, the chatbot typically has a single purpose: either the task executor, the primary locus of the interaction, or the process assistant, who can be called upon for clarification or support while the task is being executed on the GUI. When the modalities are strongly integrated, the chatbot can have multiple roles simultaneously. At each moment, the agent can act as a different operator. It can be the teacher during the user's onboarding, making a tutorial on the interface's functioning, then suggesting the user tips and tricks exploit the interface. The user can start a task by describing it to the chatbot and then modify some parameters graphically, i.e., the agent acts as a task executor. Such a chatbot can also support error recovery, guiding users through resolving the errors they are committing while using the interface. Finally, the chatbot can learn users' interaction patterns and suggest them in the following sessions or point out undisclosed functionalities of the system.

4.3 Research Questions

Having defined the boundaries of the research, we can now focus on the design of multimodal task-oriented conversational agents. We start focusing on the combination of graphical user interfaces and text-based conversational agents, but many findings are extensible to any modalities combined with a text- or speech-based CA.

First, we want to understand how introducing a new modality affects the interaction with a conversational agent. We will analyze the influence from an interaction perspective, observing how the messages sent from the user vary in a uni-modal or multimodal conversational setting, and from a design one, studying how the introduction of the second modality should influence the design of the interface and providing actionable guidelines.

Then, we will define a model to design and describe multimodal conversational interaction. Inspired by BPMN notation, we will describe the interaction as a process of tasks and gateways in which every modality is treated independently. We will create an authoring tool to enable conversation designers to graphically program multimodal CA, letting the platform transform the graphical representation into a working conversational agent.

Finally, we will implement a multimodal pedagogical conversational agent, testing the capability of the model to describe such an interface and the effectiveness of the multimodality in the final interaction.

We will analyze this domain trying to answer three research questions:

R1 How does the introduction of a second modality impacts the everyday experience from a linguistic and design perspective?

R2 How can we model task-oriented multimodal conversational interfaces

R3 is a GUI-based graphical tool an effective authoring tool for multimodal conversational agents?

The result of this work will be a 360-degree analysis of multimodal interaction, considering linguistic, interaction design, user experience, formal modeling, and implementation perspectives, producing guidelines for the design of such interfaces and a low-code tool for their implementation.

Chapter 5
Design Principles for Multimodal Conversational Interfaces

5.1 Introduction

In the previous chapters, we discussed how multimodality affects the interaction design for conversational agents. In fact, the interaction through natural language is no more *the* interaction channel but becomes *one* of the possible ones. The difference is not only in the communication channel but also in interaction possibilities this paradigm offers: the graphical user interface (GUI) requires direct manipulation, while the conversational interaction implies an indirect control [213]. Therefore, the interaction design needs to change by entering into the view of a multimodal environment early in the design process. To the best of our knowledge, very little research has been done to understand how to create multimodal conversational interfaces effectively.

This environment is the setting for our investigation. We want to examine how the design ideas for creating the best conversational interfaces described in the literature adapt or need to be changed in a multimodal environment, particularly when the conversation coexists with visual interaction. To learn how the issue of integrating a discussion with other modalities was handled, we conducted a literature review. The primary contribution of this work is a set of design guidelines from the literature research that was conducted and applied to multimodal conversational interfaces, especially those where the discussion is combined with a graphical user interface (GUI). We exemplify the resulting guidelines showing how they apply to the design of GeCoAgent.

These "heuristic" concepts were distilled from the authors' expertise in designing, developing, and evaluating various conversational applications [14, 214–221] as well as from a review of the literature. To the best of our knowledge, this is the first list of the most relevant chatbot design standards. Various writers have put forth or used a variety of guidelines, but they have yet to be standardized. Since they address chatbot-specific design principles, they can be used as a checklist to improve the usability of chatbot-specific product features from early in development and during usability evaluation - at the prototyping stage. As a result, our principles can be seen as design guidelines that complement other, more general heuristics proposed in HCI (e.g., Nielsen's 10 heuristics for inspection-based usability evaluation [222]).

The outcome of this study has been published in [18].

Table 5.1 – *Design Principles for designing of multimodal conversational interfaces*

	Design Principles
P1	Show, don't tell.
P2	Separate feedback from support
P3	Show information only when necessary
P4	Design a light interface — emphasize content
P5	Show one modality at a time
P6	Don't overload multiple modalities beyond user preferences and capabilities
P7	Use multimodality to resolve ambiguities

5.2 Design Principles

The design principles will be thoroughly explained in this section so that readers can comprehend their underlying motivations and implications.

In order to determine whether and how these principles apply to multimodal conversational agents, we first looked at the best practices and research findings for uni-modal conversational agents and multimodal interfaces in the literature.

We went about conducting our review using the PRISMA method [223]. We run the our research on Scopus, using the following query: *("design principle*" OR "guideline*") AND ("conversational agent*" OR "multimodal interface*")*, we filtered for recent papers published in the last from 2005 to 2020. The query returned 115 results. The resulting list was filtered through to identify papers that met the following criteria: *the paper's title and/or abstract must indicate that it intends to address the issue (also) from a design perspective*. 19 papers made it beyond the selection stage. We carefully read all the papers, extending our pool with relevant documents included in the bibliography, even they did not met the data criteria. The final list of consulted paper counts 29 documents.

In order to eliminate the principles, we thoroughly read the documents and categorized them by the design principles applied to the interfaces described. The seven recurring themes from this process reflect the design principles for the multimodal conversational interfaces described in the paper.

5.2.1 Show, Don't Tell.

The most direct result of adding new modalities to the interaction is the availability of more communication channels. As a result, the user can receive the information in various ways.

The agent must be created to be self-explanatory when using a uni-modal conversational interface. The discussion must include all the details required to carry on the interaction, such as the outcomes of prior activities and some suggestions for the following actions the user might take. When there are several options and detailed results, the dialogue gets verbose, lengthening messages or even necessitating more interactions to choose the desired activity, decreasing the chatbot's usefulness [224].

We draw inspiration from the well-known idea in literature to solve this issue, *show, don't tell*. The Russian playwright Anton Chekhov is quoted as saying that in narrative, arguing that things should not be stated but rather illustrated through specific examples [225]. This approach was developed based on his words. Similar to how information can be displayed across several modalities in a multimodal conversational agent as opposed to just being textually described in the discussion itself. Visual cues, for example, can guide a user through a dialogue by providing a clear summary of the actions taken and negating the need for textual descriptions [226]. A table can summarize the selections with the prior statements, and graphics can describe the data gleaned from the dialogues.

This strategy offers two benefits. The conversation's design allows for the omission of all information supplied via a different modality, resulting in messages that are both shorter and more succinct [227]. The expense of the talks can be decreased by lowering the amount of messages [224]. Second, since the dialogue uses other modalities to transmit critical information, the risk of information loss is reduced [226].

5.2.2 Separate Feedback from Support.

A conversational agent typically gives the user two types of information: support and feedback. The former shows what the user can or should do in the following interactions, while the latter includes the outcomes of the operations carried out.

This information can be delivered via various channels in a multimodal interface. For instance, the results could be displayed as graphs in a graphical user interface, the operation could be completed by changing the color of a button in the interface, and the information regarding the operations the user could perform could be written in a separate pane or included in the conversation itself.

The users should be able to identify where to look for support and where to discover the answers they are looking for, according to the Constantine and Lockwood-introduced structure principle for GUIs [228]. However, in contrast to the original approach, the separation must be constant between the interface's modules and its many modalities.

Geranium [229] is an excellent illustration of a multimodal conversational agent that uses several channels for support and feedback. An embodied, multimodal conversational agent makes up the application, which aims to make kids more conscious of the urban ecosystem. The agent probes the subject and offers commentary on the responses. When the question is posed, a set of buttons emerges, allowing kids to select the proper response. When a response is chosen, the agent's avatar plays a happy or sad animation, depending on how accurate the response is.

5.2.3 Show Information Only When Necessary.

If not appropriately used, the availability of multiple channels for communicating with users can lead to cognitive overload and a loss of usability [230].

The modalities should have complimentary content without repeating information to avoid this issue [125]. Instead of considering the dialogue as a standalone channel, we should consider it as a component of the multimodal interface. In this method, the information can be dispersed through different channels, reaching the right people at the right time with the right message. With repeats between chatbot utterances and what is on the other channels, the interface becomes clear, which makes the system less usable.

Removing the information from the interface is another concern of an excellent multimodal chatbot design. Information no longer required should be concealed to save space and reduce cognitive load.

This idea is frequently applied in those Embodied Conversational Agents (ECA) embodied where the agents' utterances are frequently recorded in balloons that vanish as the conversation progresses [231].

5.2.4 Design a Light Interface — Emphasize content.

Hearst and Tory [227] illustrate how the user's emphasis shifts to the presented data while conversing with a multimodal conversational agent, making the interface invisible to them. Consequently, a suitable design for such a chatbot reduces the total effect of the interface on the interaction. Users can only wholly focus on the conversation's subject—the action they wish to take—in this manner.

In order to adhere to this principle, interfaces must be created with a primary focus on the channel being used to transmit the information or data. For instance, if the chatbot is included on a dashboard for data visualization, much room must be given to the graphs rather than the conversation.

In the same study, the researchers observed how the user's attention would shift suddenly to the interface in cases where it was not functioning correctly or when the system provided unanticipated (or undesirable) responses, such as when the flow of a conversation was interrupted. One instance is when the conversation comes to a halt, leaving the user disappointed and the mission unfinished [171]. This effect can be lessened by carefully examining the dialogue tree and ensuring that each statement can bring the discourse to a good finish.

Ava [56] offers a good illustration of how to apply this idea by reducing the interface to just two columns—one for the conversation and one for the generated Python notebook.

5.2.5 Show One Modality at a Time.

Studies show that while people prefer multimodal interaction [125], they often only employ one modality at a time [232, 233].

The same rule should apply to multimodal interaction with chatbots, asking the user only to use one modality at a time is appropriate. Although the final work may be multimodal, the multimodality should start with alternating various uni-modal actions rather than the other way around. For instance, a conversational agent for teaching can be integrated into a visual interface where the assignments are laid out. Students can interact with the chatbot after reading the assignment to find the solution, and they can then submit the results in a different dialogue box [234, 235].

Even if all channels are not used at once, the information that gets across through the others will affect the dialogue. Since complementary information will be exchanged through the other modalities, the sentences will frequently be condensed. This idea can help with conversation design; if a process is too critical or prone to error to be explained verbally, other modalities can be employed.

5.2.6 Don't Overload Multiple Modalities Beyond User Preferences and Capabilities.

If properly utilized, multimodality can make interaction easier for the user, but if the channels are not combined in a natural and intuitive way, it will make it harder for the user to achieve their objectives [236].

Therefore, it is crucial to carefully choose the optimal channel via which the user can interact with the platform and the ones the system utilizes for sending feedback once the modalities have been created in the design phase of the conversational agent. Furthermore, similar interactions ought to use related modalities. For instance, all visualizations must be presented in a single pane, all search results must be discussed, and any potential course of action must be given through a separate list. The consistency would be appreciated by the user, who would not otherwise enjoy the interaction. [237].

The conversation history of the interface is updated each time a user or a chatbot sends a message. As a result, as the dialogue goes on, it gets longer and longer, making it more difficult to retrieve the information that was written in the messages. Because of this, crucial data should be kept somewhere other than the dialogue so that users can quickly access it.

AdApt [238] is an agent created to serve the retail industry, specifically the lookup of Stockholm apartments that are available on the market. Users have two options for the interaction: they can speak to the agent verbally or interact with a map displayed on the screen. Their Wizard of Oz study demonstrated how users utilized various channels following the system's architecture for various objectives.

5.2.7 Use Multimodality to Resolve Ambiguities.

Natural language is by its nature ambiguous [239]. These misunderstandings can jeopardize the interaction's outcome if the procedures to be carried out grow more complicated. To remedy this issue, new modalities can be added to the interface. When an ambiguity arises, the new modalities can resolve the ambiguity.

For instance, in an e-commerce website, the virtual assistant can show pictures of the product to understand the user's tastes and recommend items accordingly [208]. In end-user development, the conversation can ask the user to point out items on the screen to understand precisely what they are talking about [45]. The agent in a music chatbot can force users to listen to a brief preview of the song to ensure the one the user is referring to [215].

5.3 Case Study: Mapping the Principles on GeCoAgent

We want to exemplify the application of the design principles mapping them on GeCoAgent, the interface proposed in Chapter 2.

As said, GeCoAgent (reported in Fig. 5.1) is a multimodal interface that facilitates data retrieval and exploration for a bioinformatics application, inherently difficult activities because of the complexity of the subject matter and because they call for proficiency in search and analysis procedures, a trait common to computer scientists but sometimes lacking in biologists and medical professionals.

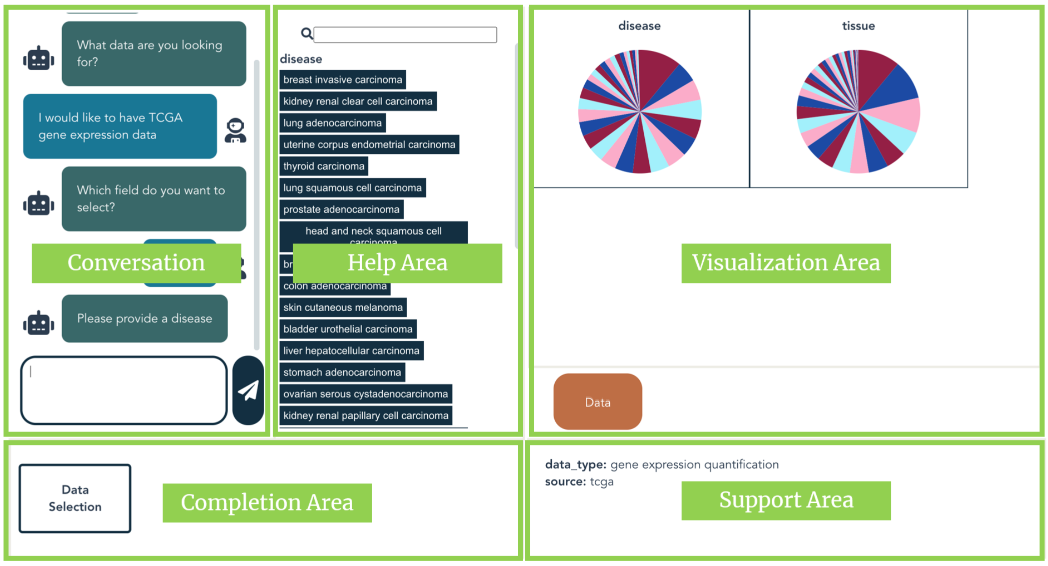
Table 5.2 summarizes how each principle has been followed in the design of GeCoAgent's interface. We provided users with two panes to get feedback from the performed operations (e.g., graphs and tables) and orientation during the process [P1]. In this way, users always know at which point of the process they are, which data are being processed and which results are obtained until that moment.

We differentiated the areas for feedback and support on the interface [P2]. Users can consult both the results of the operations and suggestions for the next moves at a glance on the same view without switching panes.

Since the tool area and the support panels have different content that changes according to the conversation (and, thus, to the operations performed), we programmed them such that their content and visualization change dynamically with the progression of the interaction [P3]. In particular, according to the feedback, the tool panel switches to the most appropriate view. Users can switch to another view by clicking on the buttons below the pane.

The interaction (and its design) with GeCoAgent is strongly driven by the data that are being manipulated. For this reason, we decided to minimize superfluous information to let users concentrate on genomic data [P4]. If users need additional information, they can ask the chatbot for further clarification, which will be provided in the chat.

Every time a message is sent to the chatbot, the chatbot replies with a textual message and some visualization. These responses have been developed so that critical information is contained only in one of the two modalities every time [P5]. For example, when users filter a dataset, the chatbot textually guides the users inserting the relevant information into the filtering operation. The GUI acts only as a support, providing suggestions to users. When an algorithm is run, instead, the textual response only suggests that users look at the graph displayed on the GUI since they are the core informative feedback.


What data are you looking for?

I would like to have TCGA gene expression data

Which field do you want to select?

Please provide a disease

disease

breast invasive carcinoma
kidney renal clear cell carcinoma
lung adenocarcinoma
uterine corpus endometrial carcinoma
thyroid carcinoma
lung squamous cell carcinoma
prostate adenocarcinoma
head and neck squamous cell carcinoma
colon adenocarcinoma
skin cutaneous melanoma
bladder urothelial carcinoma
liver hepatocellular carcinoma
stomach adenocarcinoma
ovarian serous cystadenocarcinoma
kidney renal papillary cell carcinoma

data_type: gene expression quantification
source: tcga

FIGURE 5.1 – *The interface of GeCoAgent, presented in Chapter 2. Table 5.2 illustrates how the principles apply in the interface design.*

When we designed the interface, we carefully determined which was the best modality to convey the various type of information, and therefore decided to adopt tables to show datasets, charts to visualize statistics about the data, and lists to show the available filtering options [P6].

Finally, we provide textual suggestions on all the requests to the users that might create errors [P7], such as the names of the datasets and the possible values that can be filtered in the application.

5.4 Discussion and Conclusions

More and more difficult jobs in terms of the procedure and volume of data involved are being completed using chatbots. However, more than relying solely on talk may be required and would benefit from the addition of other forms of engagement. Adding new modalities makes it easier for users who require ongoing support during interactions and can improve structured help and feedback. Even though multimodal conversational interfaces are being used more frequently, more needs to be written about optimizing their design.

We offered a set of recommendations for creating successful multimodal chatbots in this paper, which are compiled in Table 5.2. To elicit our principles, we confronted multimodal and conversational literature and clustered recurrent topics into seven indications.

These guidelines stands at the intersection between the research on usability for multimodality [125, 230] and the one for design the design of conversational agents [240–242], creating a bridge between the two worlds and trying to understand how they intersect in

Table 5.2 – *Application of the principles in the definition of the interface shown in Fig. 5.1*

	Application of each principle in the design of Fig.1's interface
P1	In the process workflow, many visuals are employed as feedback and orientation.
P2	There are various sections of the GUI with assistance and visuals.
P3	According to the conversation's situation, the GUI's information changes dynamically.
P4	The interface is built around the main components without extraneous information.
P5	Only one modality is used to display pertinent information at once.
P6	For each modality, actions and functionalities are defined.
P7	Users are helped by hints in the help section when setting parameters.

the domain of multimodal conversational agents. In fact, the findings presented in this chapter, are not to be seen as an alternative to the works in the two domains, but the adaptation of the best practices of those domains into the design of multimodal conversational agents.

We are conscious of the constraints that our work has. First, rather than being viewed as guidelines, our principles should be considered a foundation upon which the interface designer can build. Despite our thorough analysis, we only scratched the surface of this issue. Our work is the beginning of a larger conversation that includes authorities from various fields who can add to their points of view.

Our contribution is an initial effort to shed light on a mostly unexplored subject. In the next chapter, we will deepen the knowledge about the influence of multimodality in conversational agents from the dialogue perspective.

Chapter 6
How does multimodality affect the conversation?

6.1 Introduction

Before starting to design multimodal conversational agents, we want to understand in depth how multimodality affects the interaction with conversational agents. We focus on those agents with graphical elements embedded in the conversation.

Recently, with an increase in the popularity of conversational technologies in user applications, the dialogues managed by chatbots have increased in complexity, becoming longer (i.e., a higher number of messages exchanged) [243], operating on broader domains (i.e., more possibilities to choose among during the conversation) [244], supporting more complex tasks (i.e., tasks that require higher cognitive effort) [245] and providing conversations customized to the user's specific needs [246–248], such as in education and learning [249, 250], health (both mental [251, 252] and physical [253, 254]), and customer support [255, 256].

Consequently, to support the user in executing increasingly complex tasks, graphic elements started to be added to the conversational flow in natural language [257, 258] that makes interaction multimodal. These elements - hereinafter referred to as *hints* - often suggest to users what they can type to respond to the chatbot or provide a set of answer interactive options delivered in various and sometimes combined forms: concise texts, icons, or images, such as images [259]. Previous empirical studies show that this family of multimodal chatbots has a reasonable degree of usability [18, 45, 257, 260]. However, past works do not explore if the direct influence of hints introduction.

We hypothesize that the introduction of hints makes the chatbot *easier* to use, and has a measurable effect on other usability dimensions, such as interface learnability and task performance, and, more broadly, the user's conversational behavior.

Our reseach hypothesis is declined in the following research questions:

- **RQ1**: Does the presence of hints influence the interaction with the conversational agent, in terms of messages sent and time on task?
- **RQ2**: Does the presence of hints improve performance, i.e., reducing the number the errors in the interaction?
- **RQ3**: Does the presence of hints introduce effects that affect users' behaviour in the following interactions?

To answer these questions, we run an empirical study with 127 participants interacting with two task-oriented chatbots, each in two experimental conditions - with and without hints. The data we collected suggest that hints reduce the number of interactions (i.e., words), improve performance, and do not introduce learnability effects.

6.2 State of the Art

6.2.1 Multimodal Chatbots

Most information exchanged by task-oriented chatbots is textual or spoken (uni-modal) [261]. As we said in the previous chapters, prior to now, the emphasis has been on unimodal systems, which must use only one modality —spoken or textual— and still be educational and engaging [262–264]. Most dialogue systems in use supported only textual interaction, making it difficult to expand it to include other modalities like visual ones [208]. Then, with the development of artificial intelligence, the emphasis shifted. As knowledge of the informative visual modality significantly improves, dialogue systems begin to include additional modalities, including images, audio, and video, to be more robust [265]. The expressive visual modality helps text answer generation overcome challenges like describing visual characteristics [18, 208].

Among the first examples of the use of multimodality to facilitate the conversation with a chatbot, we find PUMICE and Geraneum. PUMICE [45] is a conversational agent that exploits multimodality to resolve ambiguity in the dialogue caused by the interpretation of the natural language utterances inserted by the user. However, the multimodality only applies to a few tasks the bot needs to perform. Instead, Geranium [257] takes advantage of multimodality to enhance the chatbot's knowledge and present the user with visual cues.

Liao et al. [208] created a multimodal dialogue system that can interpret many modalities as input and generates responses using knowledge of both textual and visual modalities. On the other hand, Firdaus et al. [265] offered a study of the task-oriented multimodal dialogue system, particularly in answer generation. They created an effective multimodal technique to generate a range of replies in a multimodal setup that captures data from both text and visuals. Their investigation demonstrated how the multimodal approach enhances a dialogue system's effectiveness.

Another project that deals with multimodality in dialogue systems is provided by [266]: they attempted to employ a multimodal interface to address the issue of conversational breakdowns, or system failures, to interpret the user's intentions accurately. With the mutual disambiguation pattern, which uses inputs from one modality to clear up inputs from another modality for the same notion, SOVITE hopes to solve this issue by showing the agent's comprehension [266].

6.2.2 Linguistic studies on chatbots

Language plays a fundamental role in the use of conversational technologies [267]. Several studies show how the use of language can directly impact user engagement through factors such as user satisfaction, perceived politeness, the establishment of trust, and others [268–273]. However, only some studies focus on users' perspective[267], analyzing how people use language in the interaction with conversational agents. We report the two most compelling studies.

Pickard et al. [274] study how the perceived likability of an Embodied Conversational Agent (ECA) influences people's linguistic behavior in interacting with the Conversational Agent. They discover that when the chatbot is perceived as likable, users manifest more immediacy and expressivity in the use of the language.

Second, Novelli et al. show how the communication mean (written vs. speech-based interaction) affects users' attitudes in the interaction with an embodied conversational agent. On top of that, they highlight how the difference is accentuated in people with a background in humanities [275].

These studies bring significant contributions focusing on the perception of Conversational Agents during conversations, but - to the best of our knowledge - no study quantitatively evaluated the impact of visual hints on conversational applications or the linguistic perspective.

Is it worth to mention, though, that some studies focuses on the study of cues (visual or audio) to increase the conversational agents discoverability. Examples are reported in [263, 264, 276, 277]

6.2.3 Usage of visual hints in chatbots

Multimodality is the use of different communication channels to convey content to users. In chatbot design, multimodality is often used to add multimodal items such as images, sounds, and graphical elements like buttons to the conversational interface. [18].

Studies show that even if multimodality has to be seen as a support for the user experience, users will consequently interact multimodally: they will use the modality they think fits the most for the purpose they want to achieve [125]. This modality can change at every step of the interaction. For example, in case of errors when interacting with a modality, people tend to switch to another input modality, trying to solve the problem in another way [278].

Chatbots typically reach multimodality by adding images or clickable buttons to the conversational interface. Especially in task-oriented applications, hints are meant to help the user to have a clearer idea of what the chatbot could offer. Empirical studies show that when users need to accomplish a task, the presence of visual hints helps them achieve it faster and with fewer errors [279].

Even if visual hints reportedly improve chatbots' usability, they should not obscure the possibility of using free text input. The shift to completely buttons-based results in the impression of a task environment service rather than a virtual agent [259].

In order to take the best advantage of visual hints in chatbots, some studies have been conducted to characterize them and use them in the best way possible. In their work, Valerio et al. [259] categorized them into six sign classes according to the interaction paradigm they convey or reinforce. There are:

- **Simple message**: a message that contains text and/or emojis;
- **Simple image**: a message that contains an image that could be static or animated;
- **Suggestions** or **Quick replies**: buttons that show the users what they can do at that moment; they are meant to disappear after the user clicks on them or types a message to be sent;
- **Card**: a set of actions that the user can take and, contrarily to suggestions, remains in the chat history so that the user can scroll back and choose other options from the set. A button represents each action. Buttons can stand by themselves or be associated with an image;
- **Carousel**: is a set of cards that the user can "flip through." Usually, they contain an image and text;
- **Persistent Menu**: a set of buttons that can be accessed at any time by the user.

These sign classes can be used along with different strategies. For example, graphical classes summarize the information more immediately concerning simple messages [18]. Quick replies and cards suggest the following actions that could be taken, even if the possibility of choosing from a large set of quick replies could become burdening [259].

Although many studies analyze the design of multimodal hints for chatbots and how it influences users' behavior in the application, to the best of our knowledge, no study focuses on how the presence of multimodal hints directly influences how the users converse with the agent.

6.3 Method

We conduct an empirical evaluation to see how multimodal hints affect conversational interaction in task-oriented conversational agents.

6.3.1 Goal and Research Questions

We want to analyze the influence of visual hints in the conversation with chatbots. We run an empirical evaluation through a web application to answer these questions. We asked them to complete assigned tasks by chatting with two chatbots, one after the other.

The chatbots belonged to one of the following conditions:

- *Hint* – the chatbot contained visual hints inside their interface;
- *NoHint* – the chatbot contained only text messages, without any hint.

6.3.2 Setting

The experimental procedure was self-administered. Participants were required to connect to a custom-developed web application and participate in the test in autonomy. The web platform was accessible only through connections via laptop computers.

6.3.3 Materials

The test was executed on a custom web platform, shown in Figure 6.1. The platform was developed to make participants interact with two task-oriented chatbots in both conditions (Hint and NoHint).

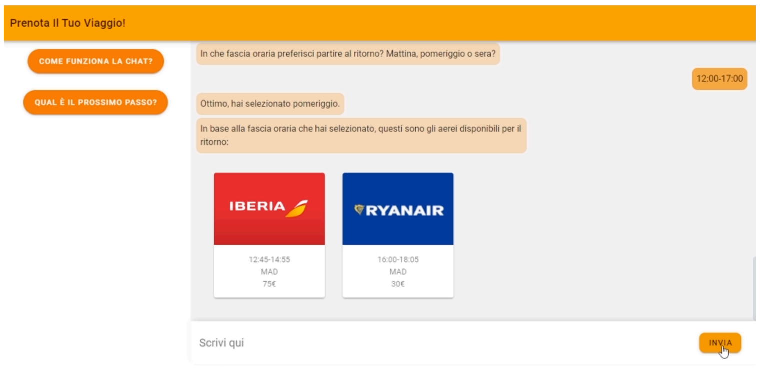
To choose the visual hints to be used for the Hint condition, we relied on the article by Valerio et al. [259], integrating it with established practices in chatbot design (e.g., those proposed by Google[1] and by ChatbotGuide[2]). The visual hints used were (Figure 6.2):

- **Quick replies:** clickable buttons were proposed to the users with different possible answers to help them choose how to continue. As soon as they clicked on one of them, they disappeared.
- **Card:** when more than one possible action was possible for the next step of the conversation, a card with the list of all options was proposed. A card could contain a set of actions the user could take, but they are persistent in the chat history even if one is clicked. In this way, the user could scroll back and choose other options from the set.
- **Carousel:** when an answer required several options from which the users could choose, each of these was proposed in a card and formed a carousel that the users could scroll to select the card they prefer.
- **Calendar and Hour Picker:** when prompted to enter a date or an hour, a calendar or an hour picker allowed the user to choose the days/hour with a simple selection.
- **Slider:** a slider is a graphical element where the user can set a value by moving an indicator, usually horizontally. It allowed users to select a range of values needed to continue the conversation.
- **Persistent Menu:** at the side of the chat, the user was offered permanent suggestions that enabled the possibility to ask for information at any time during the conversation.

Calendars, Hour Pickers, and Sliders are examples of more complex interactions, i.e., they are *widgets*. They are tools that help the user to do something like, in this case, selecting a date, an hour, or a particular value.

The two task-oriented chatbots were based on two scenarios:

[1]`https://developers.google.com/assistant/conversation-design/welcome`
[2]`https://www.chatbotguide.org`



In che fascia oraria preferisci partire al ritorno? Mattina, pomeriggio o sera?

12:00-17:00

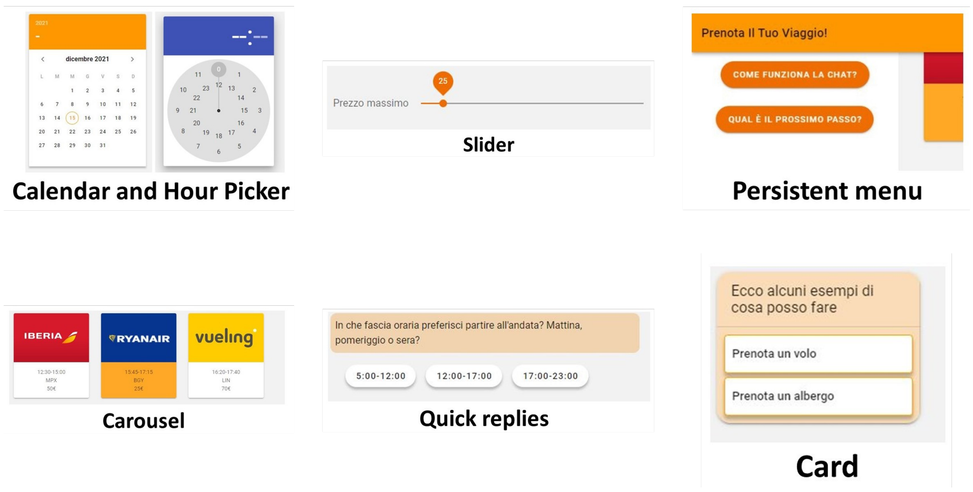
Ottimo, hai selezionato pomeriggio.

In base alla fascia oraria che hai selezionato, questi sono gli aerei disponibili per il ritorno:

12:45-14:55
MAD
75€

16:00-18:05
MAD
30€

FIGURE 6.1 – *Screenshot of the web application used for the study. In particular, the application is showing a set of cards, for the Hint condition, in the Travel Agency scenario*



In che fascia oraria preferisci partire all'andata? Mattina, pomeriggio o sera?

Ecco alcuni esempi di cosa posso fare

FIGURE 6.2 – *Examples of hints used in the chatbot for Hint condition*

Hint	Pizzeria	Travel Agency
Calendar/Hour Picker	Pizza delivery time	Days on which the travel takes place
Quick replies	Choice of the first/second type of gifts in Pizzeria	Choice of time range of the round-trip flight
Carousel	Choice of the first/second gifts in Pizzeria	Choice of the round-trip flight
Free text	Choice of the Pizzeria's location	Choice of the hotel's location
Carousel	Choice of the Pizzeria	Choice of the hotel
Quick replies	Choice of the type of pizza	Choice of the activity to do in the city
Slider	Choice of the budget for the pizza	Choice of the budget for the activity
Quick replies	Confirmations	Confirmations

Table 6.1 – *Use of the hints in both scenarios*

1. `Travel Agency`: users had used the conversational agent of a travel agency to reserve a trip from Milan to Madrid. They had to choose the period they wanted to travel, book the flights and a hotel, and ask for an activity to be done during their vacation.
2. `Pizzeria`: In the simulation of the takeaway pizzeria, users had to order a pizza. They had to choose the Pizzeria and the desired dough, and consult the menu for choosing the pizza and the time of collecting it. Moreover, two complimentary additions could be chosen to complete the order.

Although different in setting and goal, the two chatbots were created so that the tasks to be completed and the design of the conversation were as similar as possible. In particular, users had to attain a goal (order a pizza in the Pizzeria scenario and book a trip in the Travel agency scenario), answering the same number of questions and exploiting the same number of steps to achieve the goal. In this vein, we created a 1-to-1 correspondence between the questions proposed in the Travel Agency's chatbot and in the Pizzeria's chatbot, both for what concerns the type of requests and the consequent answers and for what concerns the possible hints. We designed the two conversations to exploit the highest number of different visual hints, paying attention to formulating questions that needed the same type of hints, as shown in Table 6.1.

An important design decision is that clicking on the visual hints does not correspond to sending the message directly. The text contained in the buttons, or slightly modified when dealing with calendars, hour pickers, and sliders, is copied into the typing area. To send the message, users have to click on the send button. This decision was made because we wanted to allow users to edit the text to be sent if they felt it was necessary.

6.3.4 Procedure

The empirical study consisted of a self-administered test to be completed through a web application. The test lasted around 15 minutes and was divided into two phases. The ethical committee of our research institution approved the protocol of the study.

Phase 1 – Demographic questionnaire. Users completed a questionnaire in which information such as age, gender, linguistic exposure (monolingual or bilingual), education level,

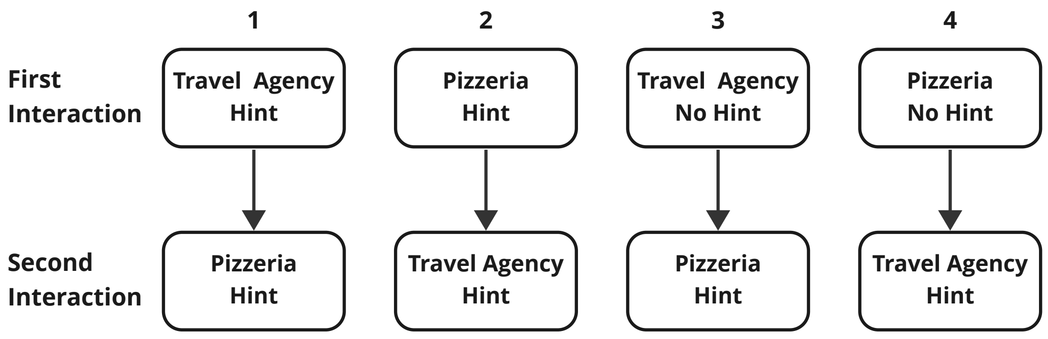


Figure 6.3 – *Graphical representation of the application's four experimental groups divided users. The assignment was done to create the most balanced groups possible regarding demographics.*

and frequency of chatbot use was requested. The collected information was anonymous: we did not collect any information from which the participant's identity could be reconstructed.

Phase 2 – Chatbot interaction. All users had to interact with the chatbots in the two scenarios to accomplish both the Pizzeria and the Travel Agency tasks. We considered the experimental session accomplished if participants completed the interaction with both chatbots. Some users had to interact first with a chatbot in the NoHint condition and then in the Hint condition, whereas others had to interact with the two chatbots in the Hint condition.

To eliminate possible bias introduced by the different conditions, we randomized the order in which the two scenarios were presented. As a consequence, study participants were assigned to one of the following four experimental groups, represented in Figure 6.3:

- *Group 1*: Start to interact with the Travel Agency chatbot and end with the Pizzeria chatbot, both with visual hints;
- *Group 2*: Start to interact with the Pizzeria chatbot and end with the Travel Agency chatbot, both with visual hints;
- *Group 3*: Start to interact with the Travel Agency chatbot and end with the Pizzeria chatbot, the first without and the second with visual hints;
- *Group 4*: Start to interact with the Pizzeria chatbot and end with the Travel Agency chatbot, the first without and the second with visual hints;

The platform used the data collected during Phase 1 to create groups as balanced as possible regarding demographics.

Before interacting with the chatbots, users had to download and read a single-page PDF file containing the instructions to accomplish the given task. Once the interaction with the chatbot was completed, the platform automatically proceeded.

6.3.5 Observed Variables

We measured the following variables:

- **number of messages sent**: the number of messages sent during the experimental session (both chatbots).
- **number of interactions:** for interaction, we mean either the typing of a single word or the click on a visual hint. If the message comes from clicking on a visual suggestion, the number of interactions is considered one. On the other hand, when the user types the message directly, the number of interactions will equal the number of words typed.
- **number of errors committed:** errors are all those things that were not expected during the interaction with the chatbot. We classify errors according to their nature, interaction, and conceptual errors. *Interaction* errors are those made by users when they type in something that the chatbot cannot understand; *conceptual* errors, on the other hand, are errors due to incorrect answers, such as requesting something that is not available.
- **time taken to complete interaction:** this is the time taken between the start and the end of an experimental session.

  All the data were stored anonymously.

6.3.6 Participants

We recruited participants voluntarily through email and social media posts. Participation in the study was completely anonymous. Participants had to be 18 years old and residing in Italy to join the study. 185 people filled out the questionnaire. We excluded those participants who did not complete the experimental sessions (both chatbots). After their exclusion, the final number of participants was 127.

The population sample comprised 73 males and 54 females, most of whom were Italian native speakers (122). The leading age group was 18-30 y.o. (46), followed by the over 60 y.o. (37), then by the 51-60 y.o. (27), the 41-50 y.o. (13) and, finally, the 31-40 y.o. (4). For what concerns the educational level, the majority of the participants completed their master's degree (60), while the rest had obtained high school licenses (32), bachelor's degrees (25), Ph.D. (6), and middle school licenses (4). The majority of the users said they never used chatbots (57) or that they used them rarely (64).

6.4 Results

In order to answer our research questions, we gathered quantitative data concerning users' interactions with the chatbot in the four experimental groups. We collected the number of messages and the average number of errors in the four groups, the number of interactions produced by the users in each session in the two scenarios (Travel Agency and Pizzeria), and the average time needed by the users to complete each session.

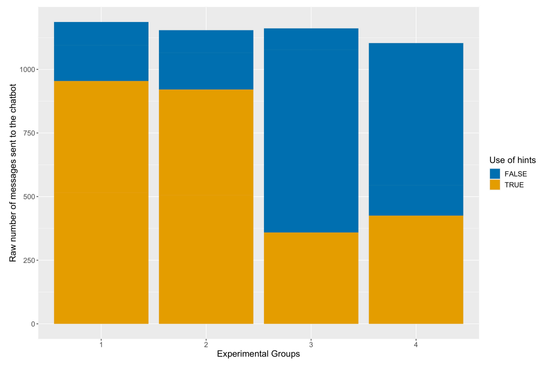


Figure 6.4 – *Raw number of messages sent in the four experimental groups. Visual hints are in orange, while written messages are in blue*

No difference emerged in the raw number of messages sent in the four experimental groups and the number of visual hints and written messages between Groups 1 and 2 and between Groups 3 and 4 (see Fig.6.4). These results confirmed that the experiment design perfectly balanced the experimental groups.

For this reason, we could perform Generalized Linear Mixed Models (GLMM) with experimental groups (1 vs. 2 vs. 3 vs. 4) and scenarios (Travel Agency vs. Pizzeria) as fixed effects and participants as the random effect to test whether (i) there was a tendency to utter fewer words when the users could select visual hints when interacting with the chatbot (both scenarios in Groups 1 and 2 and the second scenario in Groups 3 and 4) [RQ1], (ii) there was a learning effect regarding the number of words used between the first and the second scenario within each experimental group [RQ3].

Results of the GLMM are shown in tables 6.2 and 6.3. The GLMM showed that Group 1 vs. Group 2 ($p = .751$) and Group 3 vs. Group 4 ($p = .610$) did not differ in the number of interactions produced. On the other hand, Group 1 was significantly different from both Group 3 ($p < .001$) and Group 4 ($p < .001$), as well as Group 2 was significantly different from both Group 3 ($p < .001$) and Group 4 ($p < .001$).

Moving to the comparison within the condition (Fig. 6.5), in Group 1 and Group 2, no significant difference emerged between the two scenarios (both with visual hints). However, in both Group 3 ($p < .001$) and Group 4 ($p < .001$), significant differences emerged between the first scenario (with no hints) and the second one (with hints). As we can see in Fig. 6.5, in Groups 3 and 4, users were more prone to utter longer sentences in the first scenario, but once the hints were introduced, they preferred to use the visual hint option or utter a single word in the second scenario. Crucially, the average number of interactions in the second scenario of Groups 3 and 4 was comparable to those in Groups 1 and 2.

Finally, we considered the average time needed to fulfill the sessions in the four experimental groups [RQ1] and the average number of user errors in the sessions [RQ2]. As expected, participants in Groups 3 and 4 took longer to complete the sessions (on average, 11:53 minutes and 13:13 minutes, respectively) than in Groups 1 and 2 (on average, 8:26 minutes and

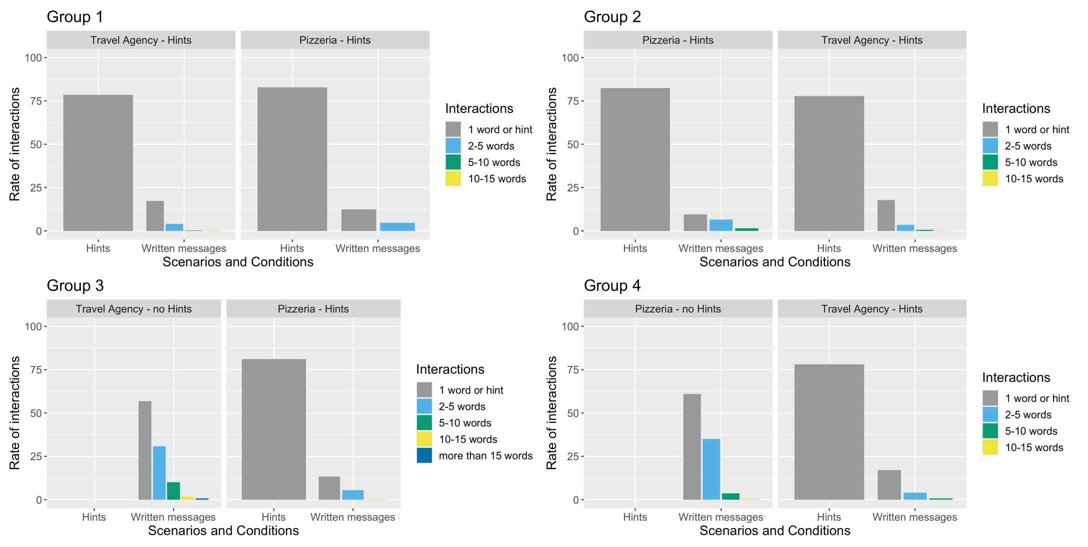


Figure 6.5 – *Rate of interactions per scenario in the four experimental groups*

7:00 minutes, respectively). Again, if we consider the two scenarios separately, the average time needed in the second scenario for Groups 3 and 4 was comparable with that of Groups 1 and 2 (on average, 8 minutes).

Concerning the average number of errors (both interaction errors and conceptual errors) in the sessions, users made significantly more errors in Groups 3 and 4 than in Groups 1 and 2 (Group 1: 6.2% of errors; Group 2: 5.8% of errors; Group 3: 12.2% of errors; Group 4: 9.3% of errors) [RQ2].

We combined data from Groups 1 and 2 to assess any learnability effect. We defined a new GLMM to find any relationship between the number of interactions and the interaction time, with interaction time being defined as a two-level fixed effect (first and second task). We randomized over the user ID and the interaction_time given the scenario. Results show no statistically significant difference in the two conditions, as shown in Table 6.4

Then, we wanted to assess the effects of the hints in the interaction. To do that, we combined groups 3 and 4 to create a new GLMM to confront the number of interactions and the presence of hints, considering the scenario and the user ID as randomized effects. Results show there is a statistically significant difference in the number of interactions in the two experimental conditions, as shown in Table 6.5

In addition, we ran GLMM to assess the possible effects of extralinguistic factors such as age, educational level, and familiarity with the chatbot of the participants on the number of interactions produced in the four experimental groups. Results showed an effect of age: participants in the older groups (51-60 y.o. and over 60 y.o.) displayed a tendency to produce a higher number of words than the other age groups, independently of the experimental condition taken into consideration. The second model showed that educational

Table 6.2 – *Generalized linear mixed model fit by maximum likelihood (Laplace Approximation)*

	Estimate	Std. Error	z value	Pr(>|z|)
Group 1 (intercept)	-3.4405	0.2168	-15.867	<2e-16
Group 2	0.2840	0.2853	0.995	0.32
Group 3	2.4105	0.2654	9.082	<2e-16
Group 4	2.1359	0.2686	7.952	1.83e-15

Table 6.3 – *Simultaneous Tests for General Linear Hypotheses*

	Estimate	Std. Error	z value	Pr(>|z|)
1 vs 2	0.2840	0.2853	0.995	0.751
1 vs 3	2.4105	0.2654	9.082	<1e-05
1 vs 4	2.1359	0.2686	7.952	<1e-05
2 vs 3	2.1265	0.2559	8.309	<1e-05
2 vs 4	1.8518	0.2593	7.143	<1e-05
3 vs 4	-0.2747	0.2243	-1.224	0.610

level modulates the number of words produced: participants with a bachelor's or a master's degree were more prone to use fewer words and to use hints to communicate with the chatbots than the other participants with lower educational levels. Interestingly, no effect emerged for the familiarity with chatbots. This result indicates that the task was sufficiently easy to perform for all participants independently from their previous knowledge of Conversational Agents.

6.5 Discussion

The goal of the present work was to study the influence of multimodal hints on the conversation.

Table 6.4 – *Generalized linear mixed model fit by maximum likelihood (Laplace Approximation), groups 1 and 2 combined*

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-4.5632	0.7653	-5.963	<2.48e-09
Interaction Time T1	0.9554	0.8010	1.193	0.233
Interaction Time T2	0.9259	0.7556	1.225	0.220
T2 - T1 == 0 (linear hyp.)	-0.02948	0.42897	-0.069	0.997

Table 6.5 – *Generalized linear mixed model fit by maximum likelihood (Laplace Approximation), groups 3 and 4 combined*

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.38442	0.09845	-3.905	9e-05
Hints	-2.63670	0.15944	-16.537	<2e-16
Hints - No Hints == 0 (linear hyp.)	-2.6367	0.1594	-16.54	<2e-16

Empirical data shows that the number of words for a turn is lower in the hint condition [RQ1]. When hints are available, users prefer to use them on the screen rather than manually typing the message in the chat. Longer messages tend to disappear, highlighting how users' preference is more robust when they have to write longer texts.

On top of that, the presence of multimodal hints decreases the cost of the conversation, not only in terms of the number of interactions but also in the time required to accomplish the task [RQ1], as shown by the comparison of the two tasks in Group 1 and 2. These factors are connected to improved usability of the system [280].

The presence of multimodal hints reduces the number of user errors [RQ2]. The presence of the cues supports users in two dimensions. On the interaction side, using graphical elements prevents the possibility of making errors while typing the text in the chat. On the conceptual side, hints support the conversation on two different levels. First, they suggest the available options - or a subset of them - users can insert as an answer to the chatbot, reducing the trial-and-error process of finding an accepted response. Second, especially when the hints do not cover the whole set of possible responses, they provide users with information about the nature of the responses the chatbot is accepting (e.g., a date, a numeric interval, a list of words, a complete sentence, etc.), guiding them on the use of the appropriate wording.

Finally, introducing hints does not affect interface learnability [RQ3]. In the second testing configuration (in which all participant groups used the multimodal chatbot), the users' performance measures do not indicate a significant difference between people who had previously interacted with hints and those who did not, and there is statistically significant difference for the interaction between the interaction without and with hints, for people belonging to groups 3 and 4. People who previously interacted with the application with hints spent less time in the interaction, and neither used a lower number of interactions than the first task. Instead, people who tried the chatbot without hints in the first scenario improved their performances, manifesting behaviors comparable to those in the other group. These phenomena show that hints are a valid instrument to flatten the learning curve of using the chatbot, supporting users to use the tool to the best of their possibilities.

Our results present some limitations. A complete study would have considered six experimental groups, the four we used plus a condition hints/no hints in the two scenarios (first pizzeria and then travel agency, and vice-versa). Yet, the estimated number of people we could reach to participate in the study would not be high enough to be able to run statistical analysis on the groups. In the future we aim at run a new, more complete study, to analyze more in depth influence of multimodal hints, considering all six the experimental conditions.

These results shed new light on how users interact with chatbots, highlighting that hints are reliable and valuable tools for helping target the conversation's goal and lightening the structure of the conversation.

6.6 Conclusions

We wanted to explore how the introduction of hints influences the interaction with a task-oriented chatbot. To do that, we run an empirical evaluation with 127 users to compare their behaviors with and without the presence of hints. Empirical results show that multimodality affects conversational interaction: users that use hints perform better in terms of time spent on tasks and errors committed and use shorter messages regarding the number of words. In addition, hints flatten the learning curve of using the chatbot.

Our study is not exempt from limitations. We tested users only on two tasks with a limited number of interactions. In addition, only metrics connected to the interaction and the number of words were considered. Future research will include a semantic analysis (NLP annotation, e.g., names, verbs, articles, adverbs, etc.) of conversational agents in order to evaluate not only quantitative data but also the quality of the conversations with chatbots and how users modulate natural language when asked to converse with Conversational Agents.

Our findings shed light on the growing phenomena of improving conversational experience through hints, leading to a more aware design of conversational interfaces. In the following chapters, we will use these considerations as foundations to systematize the design of multimodal, task-oriented, conversational interfaces through the definition of a design model and the development of an authoring tool to easily create the operating structure of such interfaces.

Part III

A model-driven approach to multimodal conversational interaction design

Chapter 7
A Conceptual Model for Multimodal Conversational Agents.

7.1 Introduction

As we analyzed in the previous chapters, in the last decade, from a user interaction perspective, we have witnessed an evolution of chatbots in various directions. Most of the first-generation chatbots were "stand-alone" and "unimodal", i.e., systems intended to be in an independent manner (*stand-alone*) and to exploit written natural language only to interact with the user (*unimodal*).

Soon Chatbots started to become "multimodal" [281, 282]: they are *inside* complex GUI-based applications and communicate with the user both verbally and through the visual elements of a GUI (Graphical User Interface) [200, 257]. As Google Conversation Design guidelines[1] state that conversation is inherently multimodal, and the design of conversational agents cannot disregard this consideration;

Our previous works, such as GeCoAgent and DSBot, predicate "complete" multimodality (i.e., both users and interfaces communicate on multiple communication channels), a new trend that is starting to be investigated. The multimodality concept is no more only a property of the system's output but belongs to its interaction paradigm.

However, in most cases, the integration between the chatbot and GUI is weak. Often, the chatbot is simply a means to offer an alternative, more natural way for the user to provide instructions to the application. In other cases, the GUI and the conversational interface are coupled but not integrated into the application: the user can perform even complex tasks using the GUI or talking with the chatbot, but the two streams of interactions are independent, and the behavior of the two interfaces is not coordinated.

In the previous chapters, we showed how multimodality and strong integration affects the design and the linguistic use of conversational interfaces and how such integration has several advantages to improve the overall quality of the user experience, particularly in applications supporting relatively multiple tasks and complex processes. We discussed how the two interfaces could complement each other to lower the cognitive barriers to using the application. The chatbot can decrease the knowledge for the user to perform the various tasks on the GUI since, during the various operations, the chatbot can provide explanations and guidance in natural language and in a contextualized way, taking into account the whole interaction history. In addition, we showed how the graphical interface could complement

[1]https://developers.google.com/assistant/conversation-design/what-is-conversation-design

conversations with information that are difficult to convey in natural language, such as data visualizations, tables, and images. Finally, the chatbot can benefit from the graphical interface's power to allow direct manipulation of interaction affordances.

Achieving strong integration between conversational and GUI-based interaction is challenging from both a design and a technology perspective. In this chapter, we concentrate on the design dimension. We propose a *conceptual model* that aims at facilitating the process of designing "strongly integrated" graphical and conversational interfaces. The model exploits the idea that the GUI and the conversational agent "predicate" on the same conceptual structure that describes the user tasks and interaction, regardless of the interaction's modality.

Our work aims to provide a tool to design the multimodal conversational interaction, not to model the interaction from a high-level perspective. Our contribution sheds light on the systematization of the design of multimodal process-intensive conversational agents. Adopting our model, an experience designer can produce an interaction model that precisely describes the final application and unambiguously describes how all the interaction modalities are exploited and what the system expects at every moment. Then, we describe the requirements a reference architecture must satisfy to support the strongly integrated multimodal integration.

The rest of the chapter is organized as follows: after presenting the current state of the art related to this topic, we will present our conceptual framework that designs an integrated interface starting from the process modeling. Finally, we will illustrate a complete case study to exemplify the use of this framework.

7.2 State of the Art

7.2.1 Design Models for Chatbots

Today, many design techniques for conversational agents exist. These techniques are linked to the conversational engine used in the application [283]. Perez et al. [283] propose CONGA DSL, a platform-independent meta-model to describe the various approaches to chatbot design. CONGA DSL's model captures the main features of chatbots, such as their name, the languages it supports, and the interaction flows composed by the dialogue between the user and the conversational agent. Together with the model, the authors implement a recommender system that, given the specification of a chatbot written in CONGA DSL, suggests the best conversational engine to build the chatbot. Still, among the dimensions captured, CONGA DSL must describe how the interaction design must happen.

In literature, there is no unique taxonomy to classify dialogue management techniques for conversational agents. Though most proposed classifications divide techniques into two leading families: handcrafted or rule-based, and probabilistic or generative based [284, 285]. The first family comprises all the dialogue managers in which the responses are designed by the developer (or the interaction designer), and the intelligent component of the system selects the best one according to the user's input. Regular structures are the most used approach for modeling the dialogue, such as instance automata [286], trees, and grammars [258]. Pauchet et al. [287] propose a computational model of human-human conversation, described through timed automata, generated through the observation of people interacting with each other to accomplish given tasks, to then use the model as input for the chatbot creation. AIML [288] and ChatScript [289] are among the most famous handcrafted conversational design techniques.

Even if outside the scope of our work, it is worth mentioning Embodied Conversational Agents (ECA), conversational agents that are embodied into a virtual agent. In this area, considerable effort has been made to understand how to describe a multimodal behavior, even if only limited to the output generation (sentences and agent's behavior). Many researchers tried to embed multimodal behavior into the dialogic representation. For example, AMPL specification extends AIML language adding the description of the facial expressions the virtual agent must perform [290]. SAIBA extends this idea to any multimodal behavior, such as expressions, gestures, gaze, lips, and body movement [291]. A similar approach is pursued by MURML, providing, together with the language specification, a reference architecture to implement embedded conversational agents [292].

Other specifications focus on modeling a language to describe agents' behaviour [293, 294]. For example, Wahlster et al. [260] proposes Multimodal Markup Language, 3ML, to model input and output for ECAs across all the modalities. Nevertheless, the authors focus on integrating different modalities in the same input and/or output modalities rather than on the interaction design.

In probabilistic models, instead, the output of the chatbot is generated by artificial intelligence, learning from examples of past conversations. Even if these methodologies are not in the scope of this paper, we mention the approach proposed by Ales et al. [295] to extract dialogue patterns from annotated data and the one from Agarwal et al. [296] to generate multimodal responses.

All these formulations, though, concentrate mainly on the definition of inputs and outputs as interaction pairs, or at most focusing on smaller conversation patterns [297, 298], with little focus on how the whole interaction takes place [299]. The reasons mentioned above make it impossible for interaction designers to describe precisely the steps of the interaction with the conversational agent. For this reason, we consider generative models out of this work's scope.

Planas et al. [300] propose a first attempt to model multimodal conversational interfaces. Their model consists of a Domain Specific Language composed of three main packages, intent, behavioral, and runtime. Using their language, the developer can design a conversational system achieving faster development and more accessible integration between the components. Nevertheless, to design the user interaction, designers must use Interaction Flow Modeling Language (IFML) [301], which mainly focuses on the system perspective rather than the interaction flow with the final user.

7.2.2 Multimodal Dialogue Systems

Traditionally, task-oriented chatbots are primarily uni-modal and based on textual information. Previously, the focus was on uni-modal systems that had to be informative and enjoyable by applying only one modality, either spoken or textual. Indeed, most of the existing dialogue systems were limited to one interaction modality, which cannot be easily extended to other modalities, such as the visual one [208]. Then, the focus changed with the growth of Artificial Intelligence. With significant advances in understanding the informative visual modality, dialogue systems have started incorporating other modalities such as images, audio, and video to be more robust [265]. Multimodality expands the expressive power of human input to computers [302]. The expressive visual modality alleviates difficulties faced by text response generation, such as the description of visual attributes [18, 208].

An example is given by John et al. [303]. They implemented a chatbot for the analysis of biomedical images. They defined a chatbot that can compute many analyses and complex pipelines understanding a controlled natural language. However, their interface is composed only of the chat and the visualization panel. The user can only chat with the bot; she cannot perform operations without using dialogue. Another example is given by PUMICE [45], a conversational agent that exploits multimodality to solve ambiguity in the conversation generated by the interpretation of natural language utterances. However, the multimodality is limited to only some specific operations required by the bot. Geranium [257], instead, exploits multimodality to complement the information provided by the chatbot and to give visual feedback to the user.

Liao et al. [208] developed a multimodal dialogue system that understands different modalities as input and uses a knowledge awareness of both visual and textual modalities to generate the responses. Instead, Firadus et al.[265] provided a study of the task-oriented multimodal dialogue system, particularly in response generation. They defined an efficient approach to generate varied responses in a multimodal setup capturing information from both text and images. Their work highlighted how this approach benefits the efficiency of a dialogue system.

Another work that deals with the multimodality in dialogue system is presented by Li et al. [266]: they tried to solve the problem of conversational breakdowns, i.e., system failures to correctly understand the intents of the user, using a multimodal interface. SOVITE wants to address this problem by visualizing the agent's understanding using the mutual disambiguation pattern, in which inputs from one modality are used to disambiguate inputs of another modality for the same concept [266]. Last but not least, Kassel et al. [304] investigate the possibility of considering the dialogue with the user as an additional bidirectional communication channel for visual analysis. Thus, they use the chatbot as a tutor for generating visualization plots.

All these researches consider multimodality focusing not on the interaction as a whole but on a subset of the interactive elements of the experience, such as the multimodal response generation, the multimodal visualization for the disambiguation, and the chatbot as a tutor for the visual modality. [200], instead, explores the possibility of embedding the chatbot in the visual interface as an alternative to the GUI-based interaction, but the chatbot is limited to be an alternative for the buttons on the screen.

In many domains, there are application-specific frameworks to integrate conversational agents (mainly speech-based ones) with other modalities. For example, commercially available home assistants have models for integrating displays in the interaction (but only as feedback) and intrinsically multimodal domains such as Embodied Conversational Agents and Automotive sectors. To the best of our knowledge, there are no studies that reason specifically on dealing with text-based chatbots and on a broader category of multimodal chatbots, i.e., the GUI and the chatbot are integrated and seen as a whole. This limits the user both in the speed of the interactions with the interface and the complexity of performed operations. Studies show that these interfaces support a more effective human-computer interaction, for example, by reducing task completion time and task error rate.

However, the natural language-based applications that provide other modalities are few, and the different modalities are often not synchronous. Taking Ava [56], a chatbot that performs a data science task, as an example, the two modalities are asynchronous. As discussed in Section 4.2.5, Ava [56] has both the possibilities of writing a Jupyter notebook directly and delineating a data science function through the natural language. However, a user cannot use the two interaction paradigms together: if she starts with the chat and then modifies the notebook, Ava is no longer aware of the changes that happened in the notebook [56].

7.3 A process-based model for Conversational Agents

In this section, we describe a model for the design of conversational agents. We focus on *task-oriented* conversational agents, i.e., agents whose interactions aim at accomplishing some user goals [189]. The goal is complete once all the intermediate objectives are complete. These objectives may be temporally independent from each other (e.g., to register to a new website, users must insert an email address and a password in any order) or with some precedence order (e.g., customers on an e-commerce platform must first choose an item and then select its color).

In Chapter 4 we introduced the following terminology:

- **Task.** A task is one of the possible goals that users want to accomplish by interacting with the system. A task-oriented conversational agent supports the execution of one or more tasks.
- **Process.** A process is defined as a sequence of (interactive) steps that describe the operations required to accomplish a task. Every task must be described by one process.

We are introducing a shift in current design approaches. When designing a task-oriented chatbot, it is common practice to start defining trees that map the dialogues. The interaction process is a natural consequence of the designed dialogue [305]. Indeed, in the design, the focus is often on the dialogue's flow rather than the process itself. For example, Botpress[2], Google DialogFlow[3], two frameworks to build chatbots, allow to delineate the possible conversation flows using a model similar to a state machine [306]. With this approach, *what* the conversational agent should do and *how* the conversational agent should do it are modeled together [307] and therefore cannot be untangled.

7.3.1 Design dimensions

Our model separates two design dimensions in the definition of conversational agents: the *what* and the *how*.

The first phase of the design process focuses on the process, describing the tasks users must perform and their precedence. In this phase, the interaction is described as a flow of high-level steps; each step represents a task that can be expanded in sub-tasks until the process is hierarchically represented in a series of atomic tasks. Only once the process has been defined does the focus passes to the detailed interaction and the messages that users and conversational agent exchange to accomplish the tasks. This clear separation of the two phases significantly benefits the design process.

First, the decomposition of the design process into smaller phases reduces its complexity. Instead of considering the whole problem as a whole, the conversation designer can concentrate on smaller, more approachable problems, having to concentrate on a single aspect of the design at a time [308]. During the process design phase, the designer can modify the process without worrying about the conversation, reducing the cost of refining the model. On top of that, logic gateways enrich the expressiveness of the process, enabling more explicit representation than in plain conversation trees.

Once the process is defined, the attention is shifted to the conversation design. In this phase, modification of the conversation does not affect the process design; therefore, the designer can concentrate on the interaction design to maximize the system's usability. At this level of granularity also, creating adaptive conversations is easier: once the goal of each atomic task is well defined, the conversation designer can think of more ways to get to it, such that the conversation can be tailored to the user's preferences at runtime.

The conversation design is progressive; first, the conversation related to tasks is defined, and then the decorative one. The design is therefore modularized; the process-related conversation that defines the essential interactions is separated from the decorative one.

The clear separation of the two phases introduces the possibility of having complementary figures working on the design, such as requirements analysts to elicit the process model and conversational design experts to take care of the interaction.

7.3.2 Modeling Constructs

We separate the design into two abstraction levels:

[2]https://botpress.com
[3]https://dialogflow.com

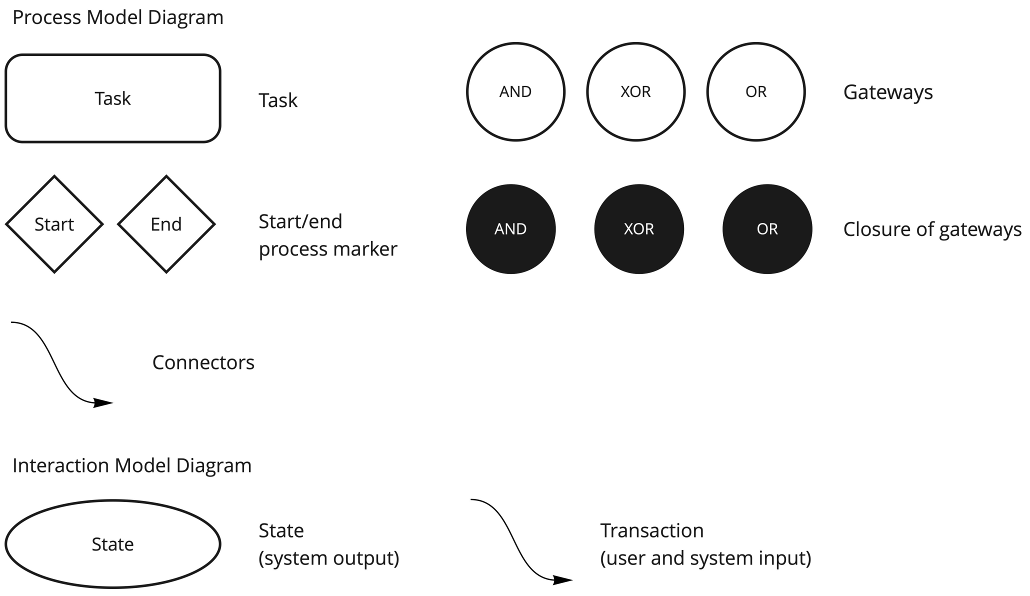


FIGURE 7.1 – *Building blocks for Process Model Diagram*

1. *Process model*: describes the procedure as a sequence of operations to be performed to complete the task, i.e., describes *what* users must do; each operation corresponds to a modification of the state of the interaction;
2. *Interaction model*: describes the interactions that each operation support, i.e., describes *how* users can accomplish their tasks; it can consist of a single step or a multi-turn interaction.

Process Model Diagram

We introduce a process notation to describe users' objectives and their precedence constraints, taking inspiration from Business Process Management Notation (BPMN) [19], a popular notation that serves well in this task, to create a state machine in which states can have two different natures:

- *tasks*: the operations that must be performed to achieve the goal of the process. We define a task as the atomic operation that influences the system's state. From the interaction perspective, a task can represent either a single-turn operation or a composite one whose sub-operations are not influencing the state, such as a form completion or a multiturn dialogue to obtain a single piece of information.
- *gateways*: describe the control flow of the process, i.e., when the user can choose how to proceed. This notation groups a list of actions that have to be performed before moving to the next steps in the process.

Figure 7.1 illustrates the graphical representation of the building blocks of the process model diagram. Tasks and gateways are linked by *connections*: represented as arrows, connections indicate the flow of the process, defining, in particular, the precedence between its activities.

The nature of the gateway determines the strategy to follow:

- *and* gateways indicate that all the actions must be performed, with no constraints on the order;
- *or* gateways indicate that at least one action must be performed, letting users decide whether to proceed with the interaction flow or traverse non-required paths before proceeding;
- *xor* gateways indicate that users must perform one and only possible path before continuing with the interaction flow.

Each gateway must be succeeded by a closure gateway that leads to the part of the process after the fork.

Each process must be delimited by a *start* and a *end* state.

Interaction Model Diagram

Interaction model diagram precisely describes how users and the conversational agent interact to accomplish the tasks the diagram represents. One and only one interaction model diagram is associated with each task and each gateway in the process model diagram. The symbols of gateway closure do not have any state diagram associated since they do not represent interaction but only rule the process flow.

An interaction diagram represents the whole dialogue related to a specific task or gateway through a state machine. Each state represents a state of the chatbot and is associated with a textual output. Arcs, instead, represents either users' intents or system events, i.e., those actions that make the flow of the interaction continue. In each diagram, there is an initial state, the state where the interaction begins, and a final one in which the task corresponding to the diagram is considered complete, and the interaction for the task terminates. In the interaction diagrams for gateways, there are multiple final states, each representing a possible transition in the process model diagram. An interaction model diagram is correct if there is at least a sequence of intents that brings from the initial to the final state of the FSM.

Despite being defined singularly, interaction diagrams represent consecutive parts of a broader process. When the dialogue reaches the final state on the interaction diagram of a task, its final state is concatenated with the initial state of the following task (or gateway), determined by the task diagram. As a consequence, the conversation continues as determined by the interaction diagram of the new active state- Any user input does not cause this progression; the result perceived by the user is a spontaneous transaction between the two states.

Adding side conversations

The formalities described so far allow a conversation designer to describe a task-oriented conversational agent. Nevertheless, the only interactions allowed are the ones that advance the process state. Such a conversational agent lacks all the typical interactions of a human-human conversation, such as requests for clarification, help support, chit-chat, and error recovery. We define these as *side* conversations.

Therefore, we introduce the description of side interactions in our model to complete the description of the conversational agent. Not all side interactions can be modeled in the same way. Some are agnostic on the state, i.e., the response does not depend on the active state, while others do. For example, the response to the user's question "What is your name?" might be the same independently from the moment the question is given, whereas the question "What should I do now?" has different answers according to the task performed.

We classify side interactions into three categories:

- *Global side interactions*. These are the interactions whose conversational agent's response does not depend on the state of the interaction. This category mainly comprises the chit-chat interaction developed mainly to give the CA a personality (e.g., *"What's your name?"*, *"What's your goal?"*) or for entertainment purposes (*"Tell me a joke"*, *"What's the whale sound?"*).
- *Task-aware side interactions*. These are the interaction whose response depends on the active task in the interaction (e.g., *"What are we doing?"*, *"Why should I do this?")*.
- *Interaction-aware side interactions*. These are the interactions whose response directly depends on the interaction that is occurring (e.g., *"What should I do now?"*, *"I did not understand"*, *"Can you repeat, please?"*.

We represent global and task-aware side interactions as separate state machines automatically inherited from all the states inside the process/task. We represent interaction-aware side interactions directly on the task state machine of the task, using a different graphical notation to differentiate it from process interactions.

All these side interactions inherit the transitions from the task state they are being issued. For example, suppose a state A has:

- a transition for the state B thanks to an intent I, and
- a transition for the side state D for an intent HELP.

Side state D has a transition to side state D', through an intent J. Suppose the interaction is in the state A, and the user sends an utterance that is classified to belong to intent HELP; the next state is therefore D. At this point, if the user's intent is classified as I, the next state is B, thanks to the transition's inheritance, despite not an explicit transition between the two states.

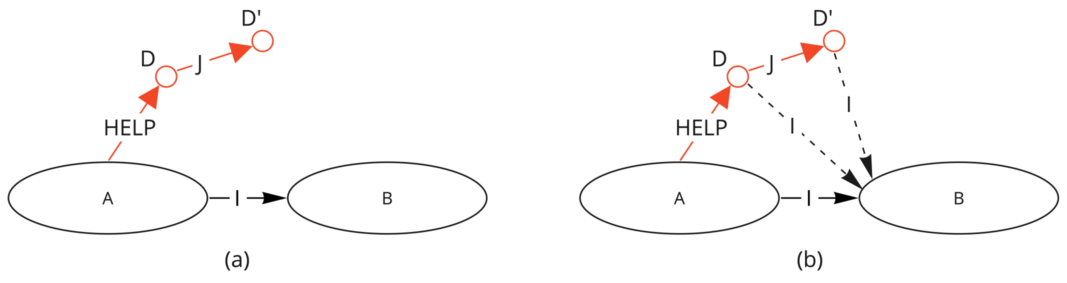


FIGURE 7.2 – *Building blocks for Process Model Diagram*

In some cases, multiple side conversations might predicate for the same task on the same intent. For example, the designer can describe a global HELP intent and a specific one for a particular task. We adopt a *local-first* policy: in case of multiple side conversations referring to the same intent, we select the one closest to the task definition; task-aware side interactions have the precedence, then task-wide, and, finally, global side interactions.

7.3.3 Design process

According to this model, the conversational design process is separated into two different moments. First, the designer chooses the communication channel of the conversational agent (text or voice) and the high-level goals of the conversation. Then, designers must concentrate on the conversation process, that is *what* the conversation should achieve to get to the overreaching goals. In doing that, the conversation process is described as a flow of connected tasks and gateways. The diagram is expanded hierarchically until the designers are satisfied with the specificity of the design.

Then, the focus is moved on *how* the interaction occurs, that is the detailed interaction between the user and the conversational agent. Conversation designers must define for each atomic task and gateway which utterances are produced by the conversational agent and which are the accepted responses from the final users. Users' messages are not unique: they must be represented as an equivalence class, identified by an id – the *intent* name – and providing a list of examples of user inputs. For each intent, we must define which are *parameters*, mandatory and optional, that the application should expect from the user and might be necessary to prosecute the interaction.

Finally, side interactions must be defined, i.e., interactions that do not contribute to the progression of the process but are essential for guaranteeing a good user experience, such as recovery error, in case the conversational agent does not understand the user's input, or help requests, or any other chit-chat interaction that enriches the experience. These interactions could be represented on the diagram as auto-loop on every task and gateway; we decide not to represent them to ease the read of the diagram.

At the end of this process, the result is a well-defined interaction flow, in the form of a hierarchical flow diagram, that precisely describes the whole application.

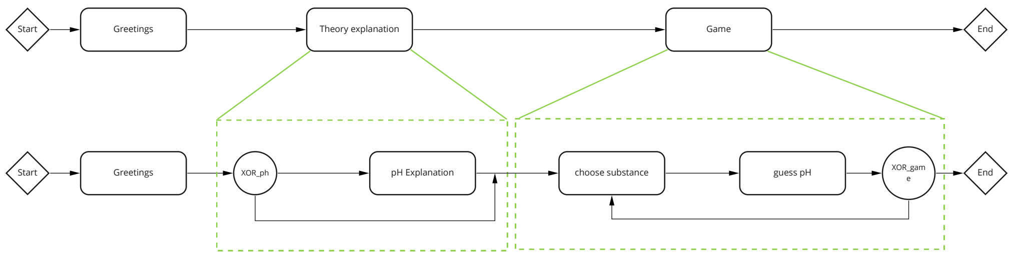


Figure 7.3 – *Task Diagram for the example provided as a case study. The lower schema is the expansion of the upper one.*

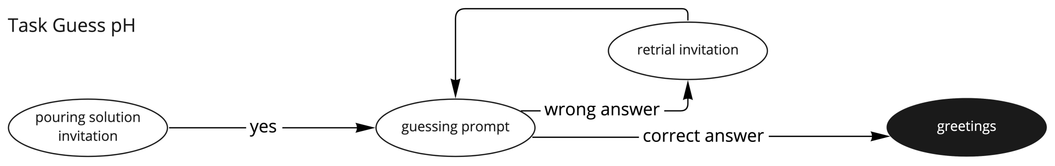
7.3.4 Example

We provide an example of a conversational agent's design to clarify our model's use better. Suppose we want to design a (text-based) chatbot to teach secondary school children chemistry, precisely the concept of pH. We want children to understand what pH is, what acid and alkaline solutions are, and how we can measure the pH of substances. While Chapter 8 describes the whole system in detail, here, we only use it as an exemplification of the model.

Following the design process described in Section 7.3.3, we start defining the high-level process of the learning act. Students must first understand pH; then, they must learn to test the pH of solutions, deducting which substances are acid and which are alkaline. We represent this process through a two-tasks process diagram model, in which the tasks `explanation` and `game` represent the operations described above. The resulting diagram is shown in the upper part of Figure 7.3.

We want to expand the diagram to represent the learning process better. We split the game task into two more concrete tasks: the user must first choose a substance to test, and then, once received the response of the pH test, guess whether it is acid or alkaline. We add some decision points that correspond to gateways into the process diagram. First, the bot asks users whether they already know the concept of pH. In case of a positive response, the explanation is skipped; otherwise the conversation continues as defined. This decision point is represented by a XOR, since only one path is taken during the conversation. Similarly, after the guess PH task, we add a XOR to let users choose to play again or quit the experience. The result is shown in the lower part of Figure 7.3.

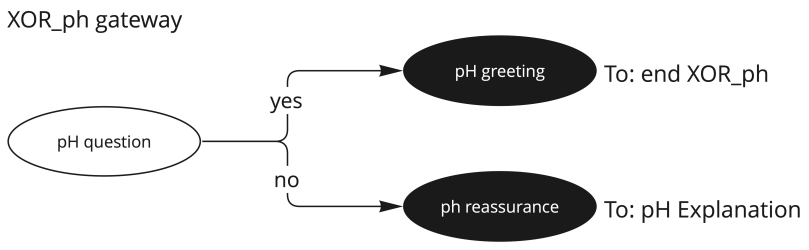
The process modeled until this step is mode independent from the interaction modality: we established what the users must do and how they must do that to continue the process. Once the process model diagram is complete, we pass to define the conversation, creating an interaction model diagram for every task and gateway that we defined. Figure 7.4 shows an example for the state `Guess pH` and Figure 7.5 the one for the gateway `XOR_ph`. For every interaction diagram, we attach two tables to describe the nature of the intent that causes the transition and the behavior of the conversational agent in every state (i.e., the sentence it has to respond to the user).



Event ID	Description
yes	User says yes to the bot
correct answer	User indicates the correct property
wrong answer	User indicates the wrong property

State ID	Output
pouring solution invitation	"Do you want to pour SOUTION_NAME into the red cabbage solution?"
guessing prompt	"Wow, the solution turned SOLUTION_COLOR! Is SOLUTION_NAME acid or alkaline?"
retrial invitation	"Oh no, try again!"
greetings	"It's right! SOLUTION_NAME's pH is VALUE, that means the solution is ACID|BASE"

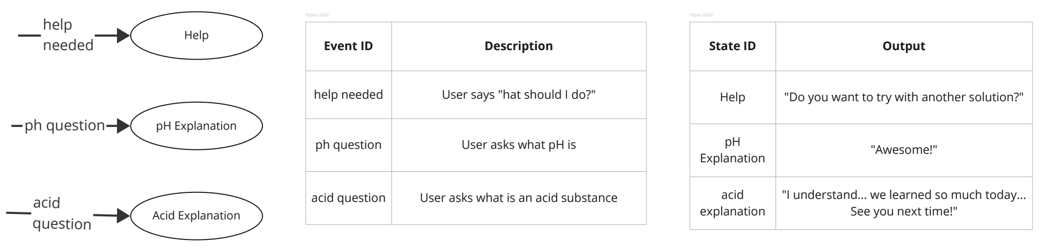
Figure 7.4 – *Example of the Interaction Diagram Model of the* `Guess_ph` *task in the uni-modal setting. The two tables underneath represent on the left the event description and the output description on the right.*



Intent ID	Description
yes	User says yes to the bot
no	User says no to the bot

State ID	Output
pH question	"do you know what pH is?"
pH greeting	"excellent, so we can play!"
pH reassurance	"Don't worry, neither I new what it was, before playing with this game"

Figure 7.5 – *Example of the Interaction Diagram Model of the* `Xor_ph` *gateway in the uni-modal setting. The two tables underneath represent on the left the event description and the output description on the right.*



Event ID	Description
help needed	User says "hat should I do?"
ph question	User asks what pH is
acid question	User asks what is an acid substance

State ID	Output
Help	"Do you want to try with another solution?"
pH Explanation	"Awesome!"
acid explanation	"I understand... we learned so much today... See you next time!"

Figure 7.6 – *Side conversations for the conversational agent.*

To conclude, we design the side conversation to create a richer user experience. We want our conversational agent to respond to questions related to the topic, such as "what is pH?" and "when is a substance acid?". These questions will be global since we want the chatbot to be able to answer these questions at any moment. Figure 7.6 shows the three conversations. They are a single-state conversation. Therefore they are represented as unique states preceded by a user event.

Once the design process is complete, we will have the complete specification of the conversational agent through a process diagram describing the high-level interaction and a set of diagrams interaction diagrams that describe the conversation punctually.

7.4 From Uni-modal to Multimodal

Even if the adoption of the proposed model shows considerable advantages in the design of uni-modal conversational agents, its major benefits lie in adopting multimodal conversational agents. Suppose we want to create a strongly integrated multimodal conversational interface, i.e., an interface where the conversational agent is not the only communication channel. However, it co-exists with other channels, such as a graphical user interface (GUI) or a virtual reality environment. We want to achieve the highest degree of integration according to the multimodality continuum presented in Section 4.2.5 Such integration brings new challenges to be faced. Before extending our model, it is necessary to analyze the possibilities of multimodal integration for conversational applications better and examine the new requirements in this scenario.

7.4.1 Requirements for strongly integrated conversational agents

Before discussing the integration, it is crucial to analyze the requirements to satisfy to reach the full integration, both from the user's and the system's point of view.

From the user perspective, we need to guarantee a smooth integration of the modalities, such that the system appears as a unique interface. As a consequence, the user's requirements are:

1. *Modality switching - ergonomy*: users must be able to switch the modality according to their preferences at every moment. For instance, a user could prefer using a form

to insert the required information to save time, whereas another wants to be guided through the process in the chat.

2. *Modality switching - confidence and security*: users must be able to use the modality they feel more comfortable and secure with. For example, a customer may prefer to insert personal information like the credit card number in a dedicated view, in spite of writing them on a chat.

3. *Transparency of synchronization*: the interface must support a switch of interaction modality at any moment. The user must be free to initiate the interaction with the chatbot and then move to the GUI or vice-versa, according to what she prefers at any moment, without worrying about synchronizing the state of the various modalities;

User requirements translate into the following system requirements:

1. *Independence of modalities*: we must be able to describe the interactions on each modality independently to have the possibility of modifying them without compromising the others.

2. *Separation of functional and interactive requirements*: to guarantee the independence of the modalities, we need to be able to modify the interaction over them without compromising the others.

3. *Support to asynchronous operation*: different modalities are not equivalent; some actions, for example, may require a unique interaction on the GUI (such as the completion of a form) but a multiturn conversation on the conversational interface, as shown in fig. 7.7. We can categorize actions to support in three families *1-to-1 interactions*, actions that are atomic in both the modalities, such as the click of the help button on a GUI and digiting *"help"* in a chat; *1-to-many* interactions, actions that are atomic on a modality, but they are not on the other, such as navigating the FAQ of a website or digiting the desired question on a chat; or *many-to-many interactions*, actions that require multiple steps on any communication channel to be completed, such as the completion of a questionnaire on multiple pages, either on the GUI or in the chat. It is worth noting that an action's atomicity is not universal but changes according to the designer's desire. For example, a chatbot that must support the user in the completion of a form might be able to see the completion of a form as a multi-step interaction, whereas if the registration form does not need the support of the chatbot, its completion and submission can be seen as an atomic operation. For this reason, we need to support transactional actions, i.e., change the state of the system, and informative actions or part of a multi-turn interaction;

7.5 Synchronizing the modalities: event-based interaction and shared context

To achieve multimodal integration, we must introduce an artifact synchronizing the modalities during the interaction. We propose to re-think the execution flow of conversational

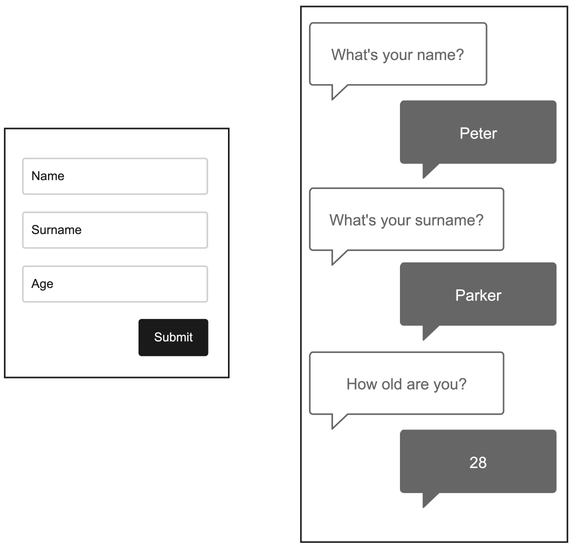


Figure 7.7 – *example of multi-turn interaction*

technologies, extending the well-established *input-intent-action-response* model: an agent receives the input, processes it, understands the intent, elaborates it, and produces a response, deciding what to say and, eventually, what to do [309, 310].

In our model, input is no more just conversational but can arrive from any active modality. We want to adopt a unified representation of all the inputs so that the system can process them similarly. Inputs are received, processed according to the modality on which they have been triggered, and converted into a universal representation. For example, both the form submission and the conversation shown in figure 7.7 would be converted into an object in the form `personal_data{name: Peter, surname: Parker, age: 28}`.

The adoption of our model, the *input-event-action-response* one, has two immediate consequences. First, all the modalities can be constantly updated on what is happening in the interface, even when the interaction is not involving them directly. Second, as we did for uni-modal conversational application, we are separating the *what* and the *how* design dimensions. Designers can modify events' triggering actions at any time, even changing modality, without affecting the process definition.

We are implicitly introducing a shift of paradigm: the application is no more utterance-based and conversation-driven, but it is event-based and context-driven, and this brings the advantage of having a multimodal interface that is synchronous among the different available channels.

7.5.1 A proposed architecture for modalities synchronization

Multimodal conversational applications need some specific elements to guarantee the synchronization of the modalities. In particular, our model requires an entity that translates

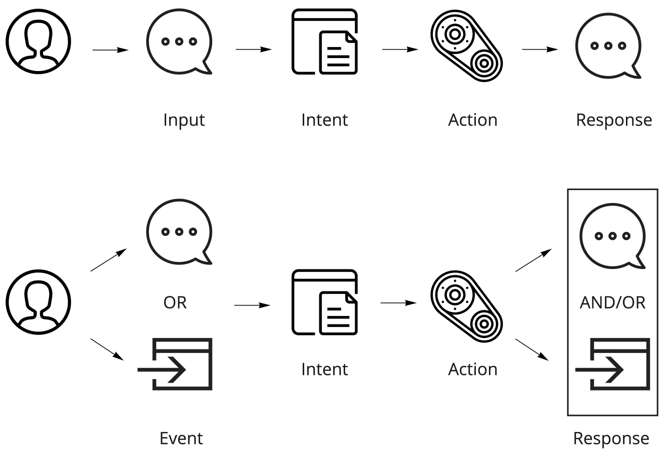


Figure 7.8 – *The shift of paradigm introduced by the introduction of multimodality: the input becomes a generic event (GUI input or utterance) and the output a combination of utterances and GUI events.*

user input into events and an artifact to store the interaction data necessary for the following interactions.

We propose a *synchronizer* module in charge to collect input from the modalities and transform them into events. The synchronizer can throw two types of events:

- *Primary*, when the event notifies that a message from a modality has arrived and the information has been stored, such as in the case of the reception of a user's personal information;
- *Secondary*, when the synchronizer must perform some computation to choose the event to throw, such as in the case of determining whether a user's answer to a quiz is correct or not.

Users' input might contain parameters necessary for the interaction's next steps. These parameters must be accessible from all the modalities. We store this information in the *context*, a shared data structure containing key-value pairs that are readable by all the modalities but writable only by the synchronizer.

7.6 Process representation as shared knowledge

We want to extend the model to exploit the context to enable the design of strongly integrated multimodal conversational agents. A first approach for the integration could be modeling the modalities as independent, with the addition of an event bus that communicates the happenings on the various communication channels.

When the complexity of the task arises, two independent models are no longer sufficient for the synchronization of the tasks: if we represent the interaction between the two modalities as processes, the transitions on both of them increase exponentially. Every state of a modality must be connected with all the states reachable from it: only in this way we can model the interactions executed on the other modalities. On top of that, in many cases, the actions executed in the states are the same, independently from the modality through which the state is triggered.

For this reason, we propose to adopt the process model illustrated in Section 7.3. The process model diagram remains unchanged; each state represents a sub-task of the interaction. This diagram is a shared representation of the process corresponding to the system's common knowledge. As in the uni-modal case, the process diagram does not describe how the interaction should occur to complete the task.

Each process model task is expanded into an interaction model diagram that declines the interaction on the modalities. These are treated separately: this diagram consists of multiple parallel state machines, one for every modality in the system. As for the uni-modal interaction, each state is associated with a uni-modal output, specifically on the modality to which the state machine is associated. On the other hand, differently from the uni-modal case, transitions here correspond to events thrown by the synchronizer. Events might trigger transactions only on a single state machine, generating a uni-modal output, or on multiple machines, generating multimodal outputs.

The FSM of each modality may have multiple final states; the interaction concludes if and only if the active state is final on every state machine. An interaction model diagram is correct if a sequence of events makes all the state machines terminate in a final state, starting from the initial one.

With this two-layered model, we simplify the design process, since we create a unique model to which the modalities refer and not a separate interaction model for each modality that participates. In addition, the separation of modalities allows the designers to modify the interaction over a modality without compromising the others. The process model makes creating and designing a chatbot application to support the different modalities' cooperation easier. It is a mechanism to generate chatbot responses, but it is crucial in the system's knowledge that the system can exploit it to interpret user input better, suggest the following actions, and answer user requests.

7.6.1 Run-time Behaviour

The runtime behavior operates as follows. When the system is initialized, the process model indicates the initial state in which the interaction must occur. The corresponding interaction model diagram punctuates the user experience: the first state of the state machine of each modality determines the initial output on every modality.

Once the user interacts, the corresponding modality captures the interaction and sends it to the synchronizer to be processed. If it is a chat message, the text is parsed into an intent, processed, transformed into an intent, and converted into an event (primary or secondary).

If the interaction occurs on another modality, it is directly processed and transformed into an event.

The event is stored and sent to the state machines of the modalities. If the received event corresponds to the event on one arc exiting from the active state, the transaction is triggered, and the output corresponding to the new state is produced. The task concludes when a transaction brings all the state machines to a final state. The control is then passed back to the process model diagram, which determines the new task of the process.

Thanks to this formulation, even if the user chooses to operate by means of the classical UI, at any moment she can switch to the chatbot, since, thanks to the synergy between the active function and the context, the chatbot is always aware of the current status of the process, even if it is involved sporadically with lots of operation between two different interactions in the dialogue.

Multimodality does not imply that the steps performed on the various modalities are mapped 1:1. For example, a form to insert personal data, that on the GUI is a unique form, can be divided into a multi-step conversation, as shown in Fig 7.7. For this reason, each state function can be conceptually divided into two parts: a first part, modality dependent, responsible for getting the input, and a second one in charge of its elaboration.

This is an important step forward compared to those systems where the chatbot has to perform the entire process. Our framework allows the chatbot to be a valid assistant to the process rather than the only means to complete it. Indeed, it can be consulted only when the user feels the need, with the guarantee that the chatbot's behavior is fully aware of the current process state. Moreover, the hard integration with a knowledge base may allow the chatbot to act in a *clever* manner, advising the user of known best practices (maybe even learned from several executions of the process itself).

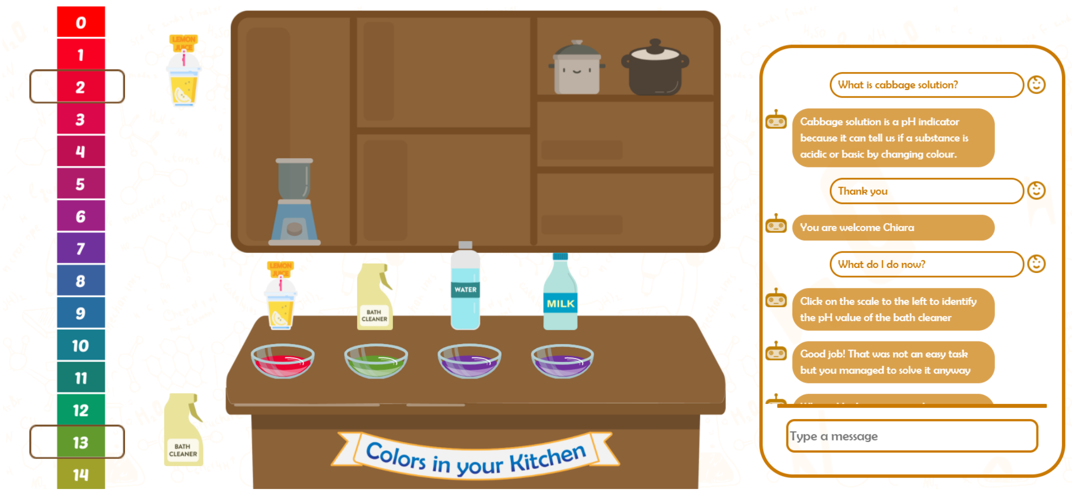
7.6.2 Example

To better explain our model, we extend the example presented in Section 7.3.4, introducing a second interaction modality. In particular, we want to create a GUI representing a kitchen counter, which is the place where the red cabbage experiment takes place. As in the uni-modal scenario, the chatbot will explain to users what pH is and guide them through the experience. With the addition of the GUI, we want children to interact through the chat or clicking the items on the screen. Figure 7.9 shows the proposed user interface.

Defining Interactions

Since our model is based on *separation of concerns* principle, the process model diagram remains unvaried. Nevertheless, we need to extend the interaction model diagram of every task and gateway to define which operations can be performed on each modality.

As said, we need to create a separate state machine to represent the interaction on the GUI. We choose to represent the interaction diagrams as separate lanes of the same diagram to facilitate the readability of the schema. Figure 7.10 shows the interaction model diagram for the guess state, i.e., the extension of the interaction described in Figure 7.4.



What is cabbage solution?

Cabbage solution is a pH indicator because it can tell us if a substance is acidic or basic by changing colour.

Thank you

You are welcome Chiara

What do I do now?

Click on the scale to the left to identify the pH value of the bath cleaner

Good job! That was not an easy task but you managed to solve it anyway

Figure 7.9 – *Multimodal version of the example provided in the case study*

The visual interface will show a kitchen counter, highlighting the solutions that users can test. Users must choose one of the solutions and drag it on the cup containing the red cabbage solution. Once the user drops the solution, an animation of the solution being poured will be reproduced, and the cabbage solution will change its color. If the user guesses the correct pH, the avatar representing the bot will make a happy face. The conversational experience remains unchanged, following the separation of concerns principle.

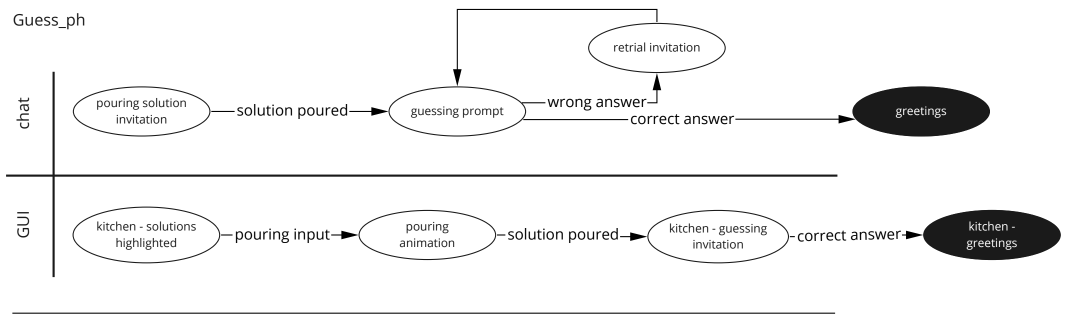
As in the case of the uni-modal design, we define in a table all the events that might trigger a state change. We add a column to the table that describes which are the modalities that can trigger each event: a user message in the chat, a user event on the GUI, or a system event, such as the end of an animation.

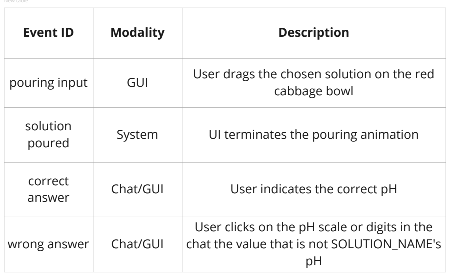
In the case of the `XOR_Gateway`, we decide to have a transition only if a `yes` event is triggered. Consequently, the initial state is also final since the transition might never occur in the interaction. The result is shown in Figure 7.11

Result: Interaction Example

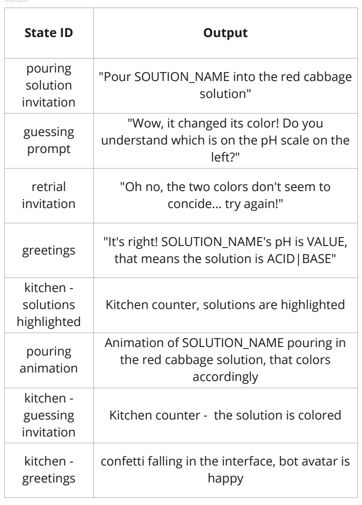
We now show an example of the interaction resulting from the application just elicited. Figure 7.12 shows a view of the interaction model diagrams of all the tasks and the gates, combined according to the specifications in the process model diagram.

Figure 7.13 shows a consequent example of interaction. The chatbot presents to the user, asking for her/his name. When the user replies by presenting her/himself in the chat, the conversational agent asks whether the user knows what pH is. At a positive response, the substances on the counter get highlighted, while the chatbot invites the user to select one of them. When the user selects the lemonade by clicking on it, the chatbot invites the user to pour that substance into the red cabbage solution by dragging and dropping it.



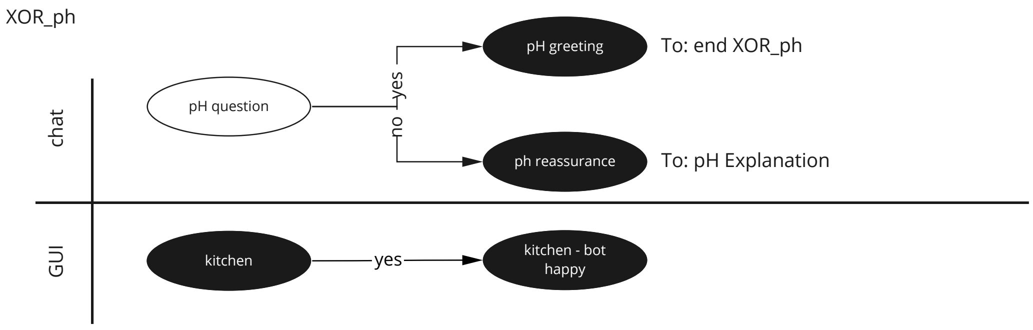


Event ID	Modality	Description
pouring input	GUI	User drags the chosen solution on the red cabbage bowl
solution poured	System	UI terminates the pouring animation
correct answer	Chat/GUI	User indicates the correct pH
wrong answer	Chat/GUI	User clicks on the pH scale or digits in the chat the value that is not SOLUTION_NAME's pH



State ID	Output
pouring solution invitation	"Pour SOUTION_NAME into the red cabbage solution"
guessing prompt	"Wow, it changed its color! Do you understand which is on the pH scale on the left?"
retrial invitation	"Oh no, the two colors don't seem to concide... try again!"
greetings	"It's right! SOLUTION_NAME's pH is VALUE, that means the solution is ACID|BASE"
kitchen - solutions highlighted	Kitchen counter, solutions are highlighted
pouring animation	Animation of SOLUTION_NAME pouring in the red cabbage solution, that colors accordingly
kitchen - guessing invitation	Kitchen counter - the solution is colored
kitchen - greetings	confetti falling in the interface, bot avatar is happy

Figure 7.10 – *Example of the Interaction Diagram Model of the* `Guess_ph` *task in the multimodal setting. Every modality has a lane in the schema. The two tables underneath represent on the left the event description and the output description on the right.*



Event ID	Modality	Description
yes	Chat	User says yes to the bot
no	Chat	User says no to the bot

State ID	Output
pH question	"do you know what pH is?"
pH greeting	"excellent, so we can play!"
pH reassurance	"Don't worry, neither I new what it was, before playing with this game"
kitchen	Kitchen counter
kitchen - bot happy	Kitchen counter - bot plays happy animation

Figure 7.11 – *Example of the Interaction Diagram Model of the `Xor_ph` gateway. Every modality has a lane in the schema. The two tables underneath represent on the left the event description and the output description on the right.*

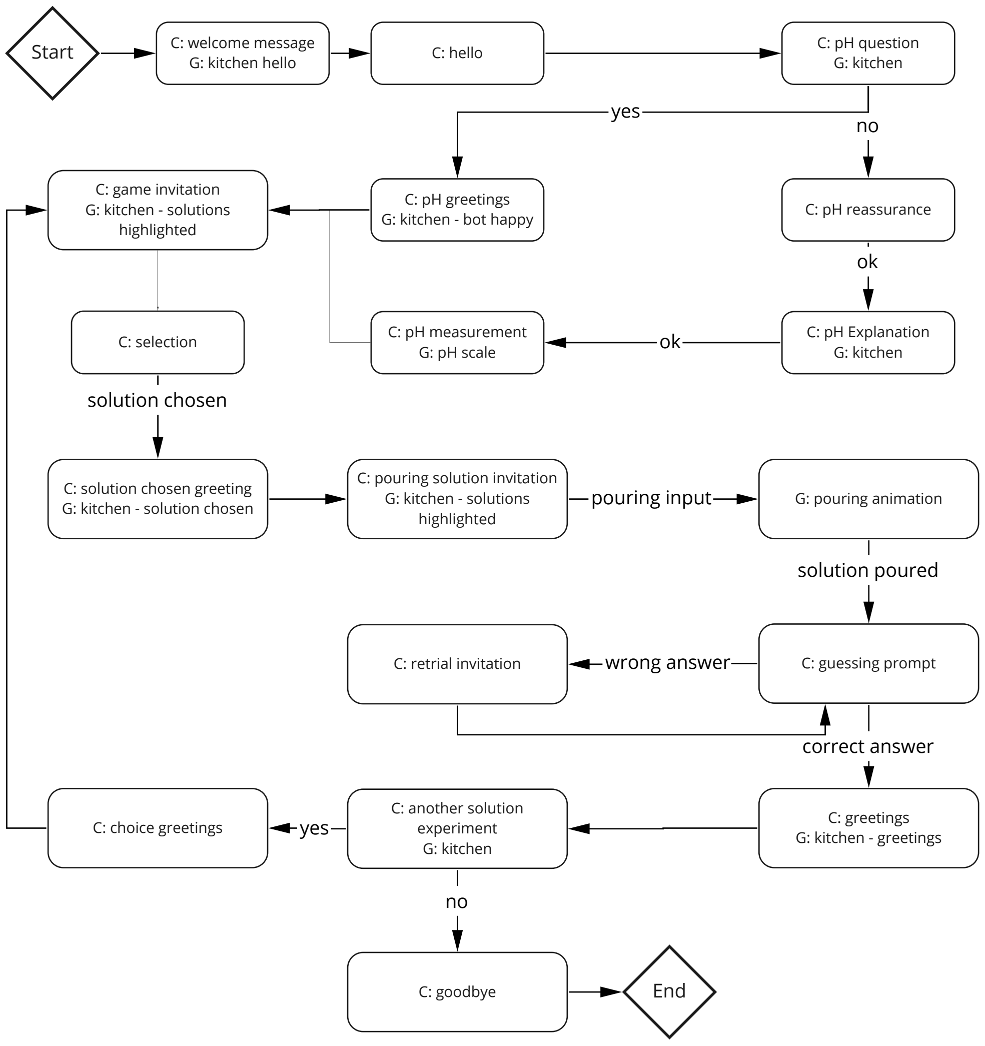


Figure 7.12 – *Process model diagram representing the expansion of all the tasks and gateways in their interaction diagram, merging the modalities.*

When the user completes this task, an animation of the lemonade bottle pouring the liquid into the red cabbage solution is reproduced in the GUI. When it finishes, the conversational agent asks the user to identify the solution's pH level, comparing the mixture's color to the scale shown on the left. When the user writes *seven* in the chat, that is the correct answer, the chatbot congratulates the user, while the chatbot's avatar makes a happy face.

Finally, when the user says she/he does not want to play another round, the avatar waves its hand to greet the user while the conversational agent sends a goodbye message.

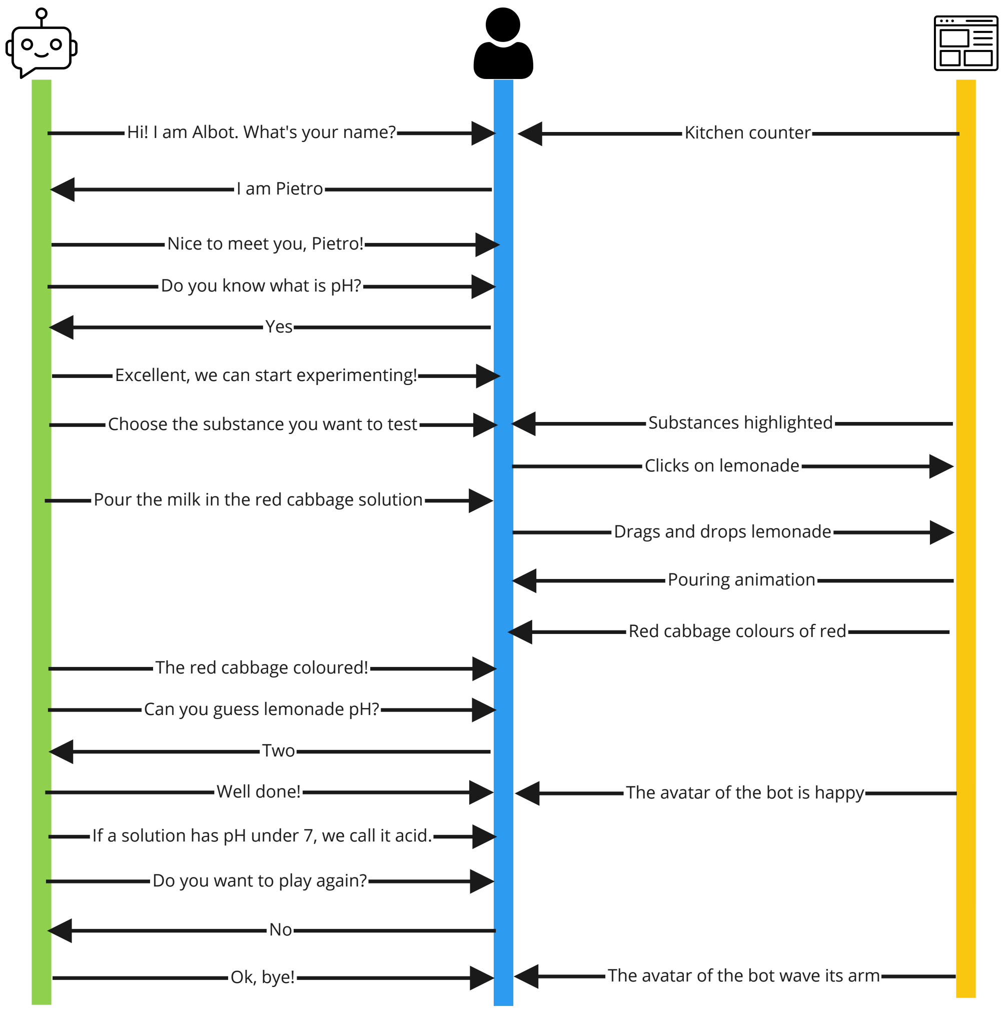

Hi! I am Albot. What's your name?
Kitchen counter
I am Pietro
Nice to meet you, Pietro!
Do you know what is pH?
Yes
Excellent, we can start experimenting!
Choose the substance you want to test
Substances highlighted
Clicks on lemonade
Pour the milk in the red cabbage solution
Drags and drops lemonade
Pouring animation
Red cabbage colours of red
The red cabbage coloured!
Can you guess lemonade pH?
Two
Well done!
The avatar of the bot is happy
If a solution has pH under 7, we call it acid.
Do you want to play again?
No
Ok, bye!
The avatar of the bot wave its arm

Figure 7.13 – *example of a possible interaction between a user and the multimodal interface of a web application teach chemistry. Particularly, on the left, there is the chatbot, on the right the GUI, and in the center the user. Each arrow pointing to the user indicates either the event or the message shown to the user, whereas the arrows pointing to the interfaces are for the message from the user to either the GUI or the Conversational Agent.*

7.7 Discussion and Conclusion

Our model is one of the first attempts to conceptualize strongly integrated conversational agents. Taking inspiration from BPMN formalism [19], we model a process as a sequence of actions that can be executed in a consecutive, predefined manner or that can be ruled by conditional logic, defined by `XOR`, `OR` blocks, or *parallel gateways*.

Such a formulation introduces a paradigm shift from a conversational-driven model to a process-driven one [196]. The conversation is now distinguishable from the model, having the process built on top, but it is created on the definition of the process itself. In this way, the dialogue is detached from the underlying process, bringing two immediate improvements [311, 312]. First, the conversation can be changed without the necessity of the model to change. Interface designers can work on the definition of the interaction, conversational and not, without worrying about compromising the task, thereby maximizing the design's efficacy. The designer can add or remove interaction steps that only exploit one of the channels without worrying about compromising the entire flow. Second, the conversation can dedicate entirely to the effective support of the user, exploiting the process model to understand what the user is trying to do and making suggestions proactively.

At the same time, our model predicates a paradigm shift in the interaction, from an utterance-based formulation to an event-based one. An event can be anything on any modality: an utterance in the conversation, a click on a GUI, audio played, or a physical button pressed on a smart object. In this way, the chatbot is not a stand-alone application anymore but is fully integrated with the other modalities in which it operates.

Finally, the role played by the chatbot changes. It becomes aware of what is happening in the whole system; consequently, it can support the user at 360 degrees, listen to the users' goals, suggest the best actions to take, and provide active support in using the multimodal application. Thus, the bot is not just an executor, a proxy for predefined actions, but becomes a tutor that follows the users in the interaction and proactively supports them.

Even if we mainly focused on written conversational agents, the proposed model is agnostic on the conversational agent's channel: both written CA (i.e., chatbots) and spoken CAs can be modeled through it. Our formulation also needs to consider how the utterances are formulated. The function in every single activity can autonomously choose how the utterance is created: features like tone, intent, and emotion expressed can be adapted to the conversation history to improve the interaction with the agent further.

Our framework is not conceived as a replacement for the conversational engines available (e.g., Dialogflow, Lex, RASA), neither for the conceptual models that describe conversational agents instances (e.g., [300, 311, 313]). Indeed, it operates at design level. Our work stresses the importance of taking into account the multimodality of the system since the very beginning of the project. We know that the underlying idea is not novel: conversation as a multimodal interaction is a widely established idea, both in disembodied and embodied conversational agents [314, 315]. However, to the best of our knowledge, the process for modeling multimodality in disembodied conversational agents has yet to be formalized.

The developer can build an interaction diagram that describes GUI and chat-based interactions through our formulation. Then, she/he can proceed with implementing the dialogic system and the user interface, keeping in mind the integration requirements.

In the future, we aim to further extend our model by introducing new modalities and policies of recommendation that keep in consideration best practices and common choices from previous interactions.

Our contribution paves the ground for a new generation of chatbots, able to support users in tasks too complex to deal with o be dealt with the conversation alone. Such agents can empower users to do knowledge-intensive tasks that today require much expertise: data retrieval, data analysis, data exploration and visualization, end-user development, and business processes are just a few examples. In this way, the new family of conversational technologies can act as facilitators to lower the learning curve of these tasks. At the same time, the tutoring capabilities of the agent can be leveraged to distribute individual users' expertise to improve the interaction's efficacy.

In the following chapters, we will see a detailed explanation of Albot Einstein, the application we used as a case study in this chapter, and a web platform to transform the model-based definition of multimodal conversational agents into the backbone of a working application.

Chapter 8
Case Study: Albot Einstein, a Pedagogical Multimodal Conversational Agent

8.1 Introduction and Research Questions

In the previous chapter, we saw the definition of a design model to define multimodal strongly integrated conversational interfaces. We presented the design of a conversational agent to teach chemistry to children to exemplify how the model works. In this chapter, we expand on this example, providing an entire case study investigating the efficacy of a multimodal conversational agent that has been produced using the model in the learning environment.

We focus on a subset of the broad family of conversational agents, namely the Pedagogical Conversational Agents (PCAs), that are the ones whose conversation has an educational goal [316]. These instruments have proven to increase students' engagement and motivation [317] and promote a more profound understanding thanks to their adaptive feedback [318]. PCAs have been successfully employed for factual learning in disciplines like safety [317], foreign languages [319], and argumentation [320]. Pedagogical Conversational Agents are also increasingly explored in STEM education, which refers to learning in Science, Technology, Engineering, and Mathematics [321].

As stated by U.S. National High School Alliance, STEM education is not only "traditional" learning of factual information but is "an approach to teaching that is larger than its constituent parts" [322]. Educators should promote the creation of connections among different STEM disciplines and facilitate the process of coming up with solutions rather than focusing on the solutions themselves only [322]. In this approach, students need to be able to experiment with problems *and* solutions in the first person, also training computational skills and critical thinking. A digitally-enhanced learning experience for STEM education can hardly employ chat-based interaction only: the rendering of concepts using multiple media and the possibility of interacting with such contents is crucial to help students explore and formulate different hypotheses based on direct observations and verify them in virtual experimental settings.

For this reason, we want to exploit the capabilities of multimodality. We believe that a strong integration of Pedagogical Conversational Agents with GUI (Graphical User Interface)-based experiences could provide a more effective STEM learning experience supporting pro-active and adaptive support to students in multiple complementary ways. In this perspective, a chatbot is no longer an independent entity but collaborates with the GUI application as part of a broader learning experience, in which users can interact in natural language with the Pedagogical Conversational Agent and in a more conventional way with the GUI in which the PCA is. This multimodal approach has significant implications for the design of PCAs, in which these agents are not the unique actors of the interaction with the user and can play multiple roles: motivator - motivating the learner and increasing engagement; tutor - guiding the learner through STEM activities (e.g., interactive experiments) and facilitating the formulation of hypotheses and solutions; instructor - teaching and clarifying theoretical concepts and correcting the student when needed; usability support - helping the user in the GUI interface use [323].

In this context, our research is born: we want to investigate the benefits of interactive applications for STEM education that adopt a multimodal paradigm - hereinafter referred to as Multimodal Pedagogical Conversational Agent (MPCA) - in which conversational interaction and GUI interaction are smoothly integrated. Specifically, our goal is declined into three research questions:

1. Can a multimodal PCA be effective for learning?
2. Does the presence of the PCA affect the engagement in the experience?
3. Which role does the PCA play in the interaction?

To answer these questions, we design, implement, and evaluate Albot Einstein, a Multimodal Pedagogical Conversational Agent, to teach chemistry, precisely the concept of pH, to middle school children. Since we already described the interaction model on which Albot Einstein has been built, in this chapter, we will focus on the pedagogical design and technological perspective of the application, Finally, through a controlled study, we empirically evaluate Albot Einstein with children (N=28) from a middle school; we compared the Multimodal Pedagogical Conversational Agent against an interactive web application providing the same GUI-based experience but not including the chatbot.

Our research offers two main contributions:

1. we exemplify how to design and implement a Multimodal Pedagogical Conversational Agent, also highlighting its underlying design principles and reflecting on the role of this kind of technology for STEM education; this articulated example and its discussion can inspire and facilitate the work of designers and developers of e-learning solutions in the STEM field
2. we report an empirical evaluation of 28 students in 2 different experimental conditions, which provides an example of a controlled study for Multimodal Pedagogical Conversational Agent that, to the best of our knowledge, is unique;

The work presented in this Chapter is currently under review at [324].

8.2 State of the Art

A wide variety of categorizations and taxonomies of chatbots can be identified due to the swift rise they experienced in the last years [325]. Pedagogical Conversational Agents (PCAs) are employed in the field of education. These agents can be defined as "lifelike autonomous characters that cohabit the learning environment creating a rich interface face-to-face with students to create rich learning interactions" [326] and have experienced a rise in employment in educational computer systems [327]. The benefits related to the employment of PCAs have been studied thoroughly and appear to be related mainly to:

- the Persona effect, which states that the perception of the learning experience of the student is positively impacted by the presence of an interactive agent [328];
- the Proteus effect, stating that students can have higher motivation if they want to resemble the agent [329].

Depending on the agent's role in the learning experience, a PCA can be classified as an instructor, student, or companion [323].

An agent is defined as an *instructor* if it assumes the role of teacher and educates the user on a defined topic [330]. Remarkable examples of the category can be found in [257, 317, 331–333]. In [257], the instructor is represented by Gera, a geranium plant that aims at teaching children about the diversity in the urban ecosystem. Different sections can be selected, and the user can interact via GUI or voice. This method allows for a more scalable and personalizable learning experience and improves a school-like interaction using technology. However, the child can grow accustomed to the method and, as it bears many similarities with regular classes, can perceive it as not engaging enough [334].

In contrast, PCAs as *students*, which are often referred to as Teachable Agents, are also a popular solution [335]. According to the "Learning by Teaching" principle, students tend to make a greater effort in studying and understanding the material when they ought to teach it to others, referred to as the Protégé effect [336]. This technique is employed in [337]: users can learn about rocks and their classification through a set of articles and pictures shown on the interface and teach the agent, Sigma, about it. The agent can request the topic to learn, and the user has to select related sentences to be taught to the agent. If they are correct, Sigma learns them, and they appear in a notebook. However, this solution does not focus on verifying the extent to which the user retains the information.

Finally, agents can also be employed as *companions* following the "Learning by Doing" paradigm [338]. In this case, the agent focuses on supporting and motivating the user. The attention is placed on the virtual environment rather than on the conversation [339]. Tutors also fall in this category. A notable example is Alcody [340], an educational environment to teach programming to Primary Education children. The users can choose the design of the interface and the companion to provide a pleasant learning environment. Children can input their programs by writing them in Alcody p-code language, which is then compiled. The agent gives feedback to the users without providing them with the right solution. Alcody also features gamification and an emotional support system. The main drawback of this class of systems lies in the lack of theoretical conceptualization of the phenomena explored in the system. Students may learn mechanically and therefore need help to generalize. For example, in [340], users were confusing Scratch and Alcody terms and modalities.

Considering the discussed points, we decided to design an agent with an intermediate role between instructor and companion (tutor). In this way, we maintained the perk of a self-paced environment with occasions to experiment while also providing detailed information on the topic to ensure effective learning.

8.3 Pedagogical Design Principles

8.3.1 Participants and Methods

We interviewed four experts, two science professors in middle school, a science communicator, and a user experience designer to elicit the design principles to follow in creating the platform, and, more in general, a multimodal Pedagogical Agent for teaching STEM.

Each expert was interviewed individually by two members of the team, one who facilitated the interview and the other who was taking notes as an observer. After a short introduction of the project, we asked them the following questions:

1. If you had to describe such a conversational agent with some adjectives, which would they be?
2. Which do you think are the strengths of such a conversational agents?
3. Which do you think are the pitfalls in which such a system might occur? How do you think we should prevent them?

Finally, we showed them some mock-ups of the interface and the dialogue, asking what they believed to be the strength of the final platform and what could be improved.

8.3.2 Results

We went through the notes taken by the observer and we found that all the answers could be condensed into six design principles.

Interactivity – Children are protagonists. Multimodal Pedagogical Conversational Agents should facilitate the execution of the experiment to children and not replace their actions. When possible, the facilitation should guarantee freedom of action to children [341], leaving them free to formulate a hypothesis and to verify them through the experiment, as suggested by the latest learning frameworks [318, 342]. In case of mistakes, the conversational agent should be supportive and encourage understanding of the error.

Clarity – Explain everything on the screen. As Graphical User Interface design principles suggest, the interface must contain only the elements that are necessary for the execution of the experiment [343]. In addition, all of them must be explained by the PCA, such that students understand what they can do to accomplish their goals. Visual cues can support the explanation, for example, highlighting the elements the chatbot is talking about.

Realism – Stick to reality. Even if the experiment is presented as a digital simulation, it must be as coherent as possible with the actual procedure so that children can learn how it works in real life [344]. The interface should not hide any step of the experiment, such that the children can learn how it works.

Safety – Try this at home. The experiment's simulation eliminates most of the risks related to experiments: in the virtual environment, students can experiment with dangerous substances and tools without the risk of being harmed [345]. Consequently, we can introduce into the application some experiences that children could not do in a real-world scenario, therefore enriching the experience. On the other hand, the application must clearly state which actions, tools, or substances are potentially harmful in a real environment.

Parallelism – Cite real-world methodologies and applications. In the experience design, the interface should explain how the concept presented is applied in the real world [346]. If the methodologies used in the experiments do not coincide with the real-world ones, at some point in the interaction, the interface should explain it or at least mention it.

Rythm – Fragment long explanation in multi-turn interactions. The PCA should avoid long explanations as much as possible. If necessary, the conversation designer should fragment them into a dialogue interrupted by interactive moments such as direct questions to the user to keep the engagement high [298].

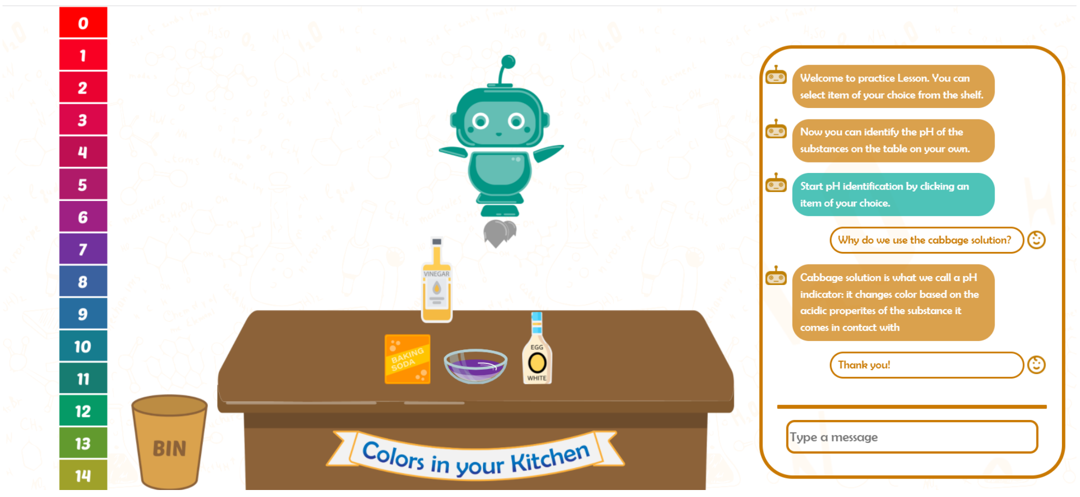
8.4 Albot Einstein

8.4.1 User Experience (UX)

Albot Einstein is a multimodal Pedagogical Conversational Agent to introduce middle school children to chemistry. By experimenting with the e-learning experience, students can discover the concept of pH, what acid and alkaline solutions are, and how to measure the pH of a substance.

As shown in Figure 8.1, the experience is set in a virtual kitchen; on the counter, children find common solutions (e.g., soda, vinegar, milk, lemonade, egg yolk, etc.) and a bowl containing red cabbage solution, the pH indicator used in the experiment. The visual interface is coupled with the multimodal PCA, which guides children through the experience. We will refer to the agent as PCA or chatbot interchangeably. A pH scale on the left completes the interface; it references the colors assumed by the red cabbage solution when exposed to acid or alkaline solutions.

Students interact with Albot Einstein by clicking on the Graphical User Interface or writing textual messages in the chat. Some interactions require modality-specific actions (e.g., *click on the lemon juice button to pour it into the red cabbage solution*), whereas others can be executed on any modality (e.g., children can guess the pH level of a substance either clicking it on the pH scale or writing it in the chat). The interface is fully multimodal: the



Welcome to practice Lesson. You can select item of your choice from the shelf.

Now you can identify the pH of the substances on the table on your own.

Start pH identification by clicking an item of your choice.

Why do we use the cabbage solution?

Cabbage solution is what we call a pH indicator: it changes color based on the acidic properites of the substance it comes in contact with

Thank you!

Figure 8.1 – *A screenshot of Albot Einstein's interface.*

two modalities are aware of what is happening in the counterpart. In this way, for example, the interaction in the GUI can proceed even if the assigned task is completed through the chatbot, or the chatbot can highlight some elements in the interface to clarify what it is talking about.

Albot Einstein consists of two activities, introductory mode and free game one. In the introductory mode, children discover the concepts related to pH. To do that, the PCA guides step-by-step in measuring the pH of four substances, an acid, two alkalies, and water, by pouring them into the red cabbage solution and then confronting the color of the solution with the ones on the pH scale. Proceedings with the experiments, the chatbot questions users to stimulate their critical thinking, inviting them to formulate hypotheses on what they are observing while introducing the concepts of pH scale and indicators, acid and alkaline solutions, and strong and weak acid/alkaline solutions.

In the meantime, children learn to interact with the platform through all the modalities. When children complete the task, the free game starts. A set of testable solutions appear on the counter, and the child can test their pH level independently. In this phase, the PCA occasionally stimulates children's curiosity by asking questions that can be answered by interacting with the experiment (e.g., "Which substance is more acid? vinegar or lemon juice?").

At any time, if children have any doubts or curiosity, they can directly ask the chatbot, which provides further explanation.

Table 8.1 describes how the elicited principles (Section 8.3) have been implemented in the design of Albot Einstein

Table 8.1 – *Implementation of the elicited design principles (described in Section 8.3) in Albot Einstein*

Design Principle	Implementation
Interactivity	All the explanation moments follow interactive moments in which users can formulate and test their hypotheses
Clarity	At the beginning, the PCA presents all the interactive elements on the screen and their use
Realism	The proposed experiment is a loyal reproduction of a real activity
Safety	Children can test dangerous substances such as bleach risk-free
Parallelism	Real-world instruments used to measure pH are described during the experience
Rhythm	Longer explanations are fragmented with questions, practical examples, puns, etc.

8.4.2 Runtime Model

A platform able to guarantee such an experience presents three main requirements. To be multimodal, the system has to process input from both communication channels synchronously: the chatbot must be aware of what is happening on the interface and vice versa. There needs to be more than synchronous communication; to be proactive, the chatbot must take the initiative and send content to the interface through asynchronous communications. Finally, the experience must be flexible and adaptable with little effort, such that new elements to identify or dialogues can be introduced when needed.

The backend architecture is built around two main modules, the Natural Language Processing (NLP) Module and the Dialogue Manager, for the system to process simultaneous inputs. The Dialogue Manager is responsible for conducting the interaction. Figure 8.2 exemplifies its functioning. The Dialogue Manager listens for input events, both textual (in the chat) and graphical (on the GUI). In the first case, it sends the message to the NLU module to identify the user's intent and parameters, if any. To parse users' messages we use RASA, an open-source Natural Language Understanding Unit [120].

From this point, textual input and interface interactions are generic events and therefore are treated in the same way. Then, it processes the input is processed to produce the desired output. The output is also multimodal: the dialogue manager produces both text messages for the chatbot and modifications on the GUI.

The dialogue manager bases decisions on a state machine that formally represents the designed experience. A JSON describes the state machine; it specifies for each state a list of possible inputs, and for each input:

- the chatbot's utterances;

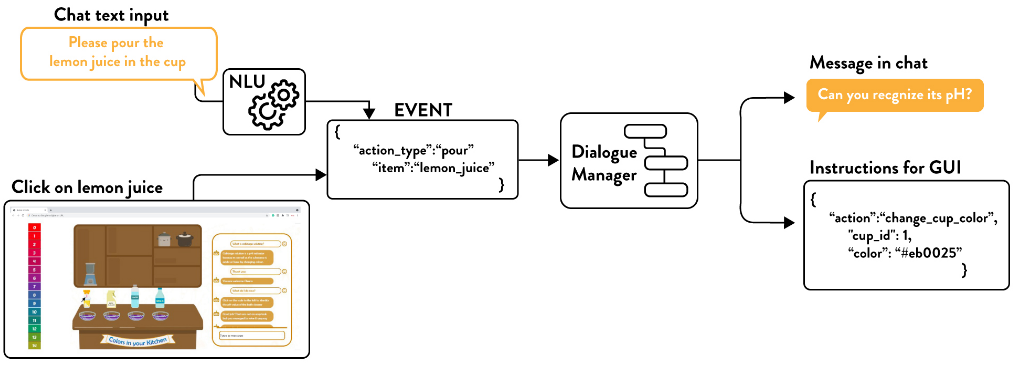

Chat text input
Please pour the lemon juice in the cup
NLU
EVENT
{ "action_type":"pour" "item":"lemon_juice" }
Click on lemon juice
Dialogue Manager
Message in chat
Can you recgnize its pH?
Instructions for GUI
{ "action":"change_cup_color", "cup_id": 1, "color": "#eb0025" }

Figure 8.2 – *Runtime model of Albot Einstein. Text input in the chat is converted into events, and the interaction is captured on the GUI. The dialogue manager processes those events thanks to its process representation and produces the output for all the modalities.*

- the modifications of the GUI, in objects displayed and phases of the game;
- the next state of the user, given the input.

The design is flexible since the experience can be quickly modified only by acting on the JSON. The communication system between the backend and the front-end leverages a custom WebSocket implementation to provide proactive interaction from the PCA.

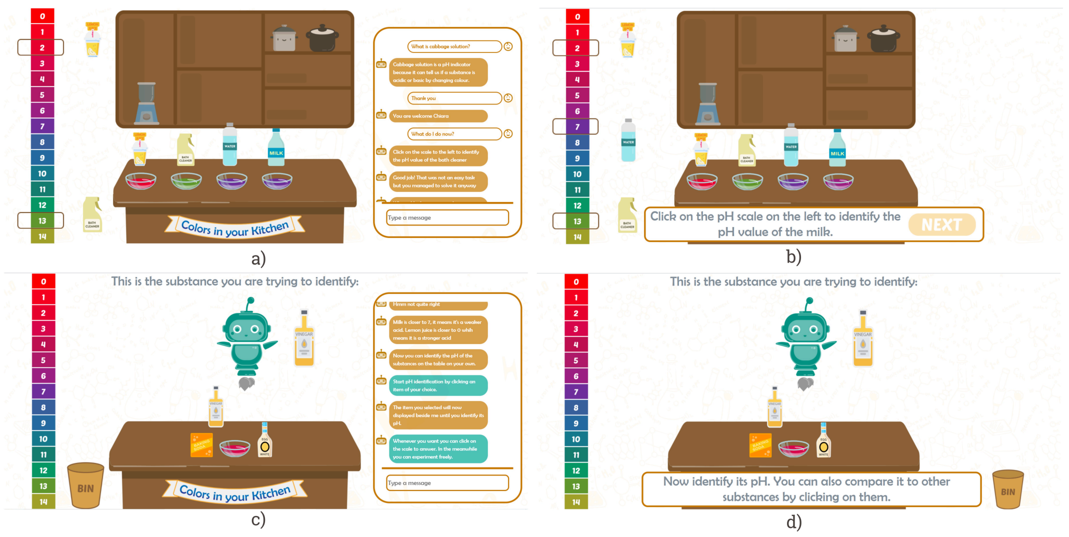
8.5 Evaluation

8.5.1 Goal and Research Questions

We want to explore the potential of Multimodal Pedagogical Conversational Agents for teaching STEM subjects, particularly chemistry.

To answer these questions, we run an experimental study with 28 children in a middle school. These children have never studied the concept of pH at school. To assess the role that the multimodal Conversational Agent plays in the application, we divided participants into two experimental conditions:

1. *Chatbot (C):* participants interact with the multimodal Pedagogical Conversational Agent (Fig. 8.3 a,c);
2. *Interface (I):* participants interact with a similar application, but the instructions provided by the chatbot are replaced by textual prompts on the interface (Fig. 8.3 b,d), allowing learners to interact only through the Graphical User Interface.



What is cabbage solution?

Cabbage solution is a pH indicator because it can tell us if a substance is acidic or basic by changing colour.

Thank you

You are welcome Chiara

What do I do now?

Click on the scale to the left to identify the pH value of the bath cleaner

Good job! That was not an easy task but you managed to solve it anyway

Type a message

Click on the pH scale on the left to identify the pH value of the milk.

Hmm not quite right

Milk is closer to 7, it means it's a weaker acid. Lemon juice is closer to 0 whih means it is a stronger acid

Now you can identify the pH of the substances on the table on your own.

Start pH identification by clicking an item of your choice.

The item you selected will now displayed beside me until you identify its pH.

Whenever you want you can click on the scale to answer. In the meanwhile you can experiment freely.

Type a message

Now identify its pH. You can also compare it to other substances by clicking on them.

Figure 8.3 – *Application in the two experimental conditions. The GUI is integrated with the Pedagogical Conversational Agent on the left-hand side (a, c). Learners can interact directly by manipulating the interface and through text messages in the chat. On the right-hand side, the PCA is replaced by textual information displayed at the bottom of the screen.*

8.5.2 Participants

We recruited 28 children from a middle school. All participants were aged between 10 and 14 (M=12) and balanced in gender (14 F, 14 M). The participants from each group did not differ on variables like gender, average age, and class they belonged. Participants were selected from classes that had not previously studied the concept of pH. All the participants joined the study voluntarily. Before the study, both the students' parents signed a consent form that explained the study's purpose and procedure and included all regulatory rules we followed to guarantee the anonymity and privacy of the data collected. Before starting the experimentation, children had to pick up a nickname to use in the activity, such that no personal data was stored. Additionally, according to the assigned experimental condition, each participant was assigned an identifier, C1-C14, and I1-I14.

8.5.3 Setting

Participants were met in a room individually, with the same setting for the two experimental conditions. Two members of the research team were present during the study, one to guide the user in the experimentation process (i.e., the *facilitator*) and one to take notes on the interview (i.e., the *observer*). Participants sat in front of a laptop on a table to interact with the web application, with the facilitator next to them. The observer was also sitting in the room with a laptop to take notes. In addition, the participant's laptop was sharing the screen with the observer, so the observer could watch the interactions on the screen without interfering in the activities.

8.5.4 Methodology

Each participant was met individually in a semi-structured interview. The interview was structured into four phases and lasted around 15 minutes. The ethical committee of our research institution approved the protocol of the study.

Phase 1 – Preliminary assessment. Before the beginning of the interview, participants were asked to choose a nickname among famous scientists' names. Subsequently, students were required to answer an assessment questionnaire. The questionnaire consisted of 10 closed questions that investigated students' prior knowledge of pH.

Phase 2 – Interaction with Albot Einstein. Once participants completed the preliminary assessment, the facilitator introduced them to Albot Einstein. According to the experimental condition they were assigned, the platform interface was different:

- Chatbot (C): the GUI was integrated with the Pedagogical Conversational Agent (Fig. 8.3 a,c). Participants could communicate both through the Graphical User Interface and the PCA;
- Interface (I): the PCA was replaced by a textual prompt at the bottom of the GUI. The textual prompt showed the same information as the chatbot in the (C) condition, but learners could not interact textually with it (Fig. 8.3 b,d).

In both conditions, the facilitator asked participants to follow the instructions given by Albot Einstein to accomplish two tasks:

1. T1: Complete the introductory mode;
2. T2: Successfully individuate baking soda, egg white, and vinegar pH.

The facilitator did not give any instructions not to introduce bias in the data. Observer measured:

- Number of errors: errors committed by participants; errors were classified as *Interaction errors* when users use the interface not as intended, and *Guessing errors*, when the users choose the wrong pH value on the scale;
- Requests for help: number of times the participants asked the facilitator for help. Requests were classified into *Interaction requests* if participants asked about how to use the interface, and *Clarification requests*, if participants asked about the didactic content of the application.

At the same time, Albot Einstein logged:

- *Time on task*: the time spent executing task 1 and task 2; defined as the time elapsed since the complete loading of the application page until the appearance of the completion message;
- *Number of messages*: in the chatbot condition, Albot Einstein logged the number of messages sent by the user to the chatbot in every task.

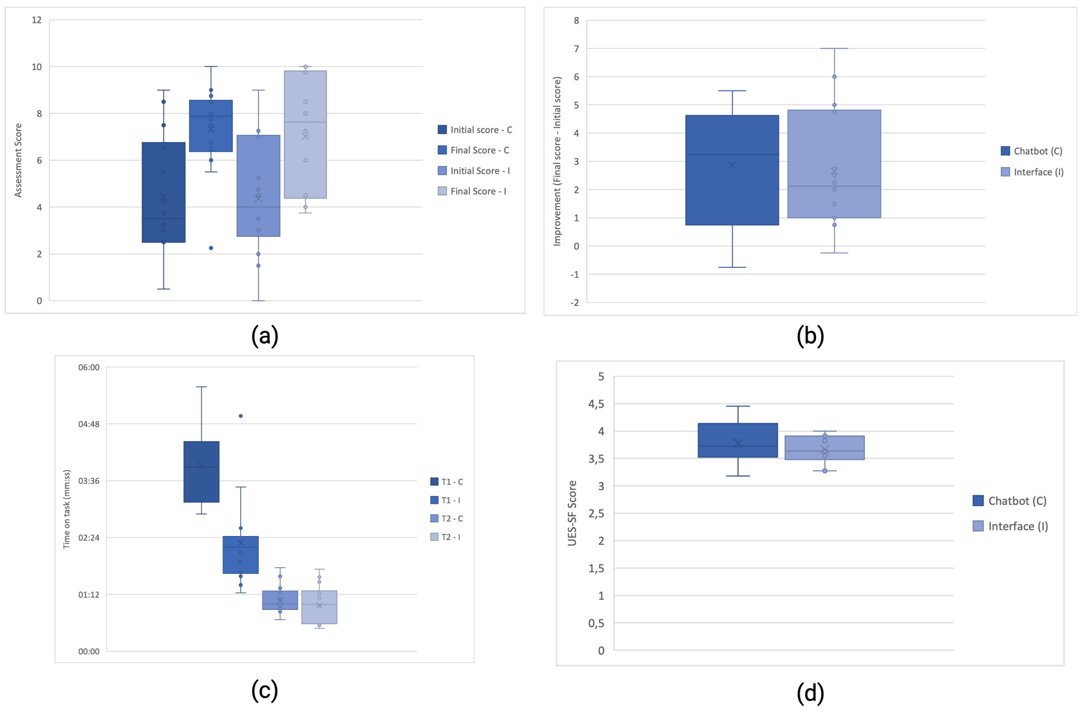


Figure 8.4 – *(a) distribution score of the assessment test before and after interacting with Albot Einstein. (b) improvement of users' scores in the assessment test, computed as the difference between the final and initial tests. (c) time on task for the two tasks in the two experimental conditions. (d) UES-SF scores in the two experimental conditions.*

Phase 3 – Assessment and User evaluation. Participants were invited to complete an assessment questionnaire to verify the knowledge acquired in the interaction with Albot Einstein. The questionnaire was the same administered during phase 1.

Then, participants had to complete UES-SF (User Engagement Scale - Short Form) [347] questionnaire to measure their engagement with the application.

Phase 4 – Final Interview. The session concluded with a semi-structured interview to qualitatively understand how users perceived the version of the platform used in their experimental condition. The questions were:

- What is the thing you liked the most?
- What did you not like about Albot Einstein?
- Do you think Albot Einstein is useful for your education? Why?

8.5.5 Results

Competences assessment

All 28 of the participants successfully completed the study. As shown in Figure 8.4(a), before playing with Albot Einstein, participants in Chat condition (C), scored on average 4.44 points out of 10 ($N = 14, SD = 2.55$), whereas in Interface condition (I), the average score was 4.35 points out of 10 ($N = 14, SD = 2.56$). In both conditions, the scores normally distribute according to the Saphiro-Wilk test ($C : W = 0.916, p = 0.893; I : W = 0.967, p = 0.862$)

On average, the second test scored 7.32 points out of ten in C condition ($SD = 1.90$) and 7.09 points in I condition ($SD = 1.90$). Also in this case, according to the S-W test, the conditions distributed normally ($C : W = 0.916, p = 0.893; I : W = 0.967, p = 0.862$)

We run a Repeated Measures ANOVA on the scores and we found that the difference in the performances between before and after interaction was statistically significant ($F = 45.84, p < .001$). Yet, there is no statistically significant difference in the performance of the groups in the two conditions ($F = 0.07, p = .795$).

Interaction

As shown in Figure 8.4(c), on average, students in C condition spent on average $3m54s$ executing task T1 ($N = 14, M = 3m54s, S.D. = 49s$), and $58s$ executing T2 ($N = 14, M = 58s, S.D. = 25s$). T1 timings are distributed normally according to K-S test ($D(13) = .14, p = .2$), whereas T2 ones are not ($D(13) = .24, p = .03$).

Students in I condition, instead, spent on average $2m17s$ executing task T1 ($N = 14, M = 2m17s, SD = 25s$), and $1m04s$ executing T2 ($N = 14, M = 1m04s, SD = 18s$). Both the distributions are not distributed normally ($T1 : D(13) = .25, p = .01, T2 : D(13) = .20, p = .03$).

Participants in condition C were significantly slower than the ones in I condition in executing T1, as shown by Mann-Whitney U test ($U(N_C = N_I = 14) = 18, z = -3.68, p < .001$), whereas the same test did not find a significant difference in T2 timings ($U(N_C = N_I = 14) = 80.5, z = -.81, p < .42$).

No participant committed any Interaction error that was unable to recover by themselves. All the requests (both interaction and clarification) for the facilitator were solved in autonomy by a more careful analysis of the application. Instead, participants made on average 1.21 and 1.64 guessing errors respectively in C and I conditions ($C : N = 14, M = 1.21, SD = .80; I : N = 14, M = 1.64, SD = 1.28$). None of the two distributions is normal according to the K-S test ($C : D(13) = .27, p = .01; I : D(13) = .34, p =< .001$). Again, no significance was found in the difference between the two distributions by Mann-Whitney U test ($U(N_C = N_I = 14) = 86.5, z = -.57, p < .57$).

Participants interacting with the multimodal PCA sent from 4 to 7 messages per person ($N = 67, M = 4.77, S.D. = .89$) during T1, whereas only one participant sent messages in the second part (P11, $N = 2$).

Table 8.2 – *Users' answers to the question "What did you like the most?"*

Liked Aspect	Conversation (C)	Interface (I)
Fun experience	0	3
UX/UI	2	6
Explanations	1	3
Task T2	2	1
Chat	9	N.A.

UES-SF questionnaire data

All the participants completed the UES questionnaire after interacting with the platform. The overall score, obtained as the average score of every question, is 3.78 in C condition ($M = 3.78, S.D. = .39$) and 3.66 in I condition ($M = 3.78, S.D. = .25$) (Figure 8.4(d)). Both distributions are normal according to the K-S test ($C : D(13) = .17, p = .2; I : D(13) = .14, p = .2$). Though, there is no significant difference between the two distributions ($t(26) = -.94, p = .35$).

Qualitative interviews

We analyzed the data coming from the semi-structured interviews by running a thematic analysis [119].

At the question *"What did you like the most?"*, in C condition, seven participants said the chat's support and its role in the game, two said the second activity (T2), in which they could identify the pH in autonomy, one the explanations inside the activity, and two the interface and experience design. In I condition, instead, five people appreciated user experience and interface design the most, three the explanations inside the game, three the fun they had while playing with Albot Einstein, and one the second task.

In general, Albot Einstein was described as interesting [C1, I4, I9] and enjoyable [I1, I3, I5]. Nine participants highlighted that the experience was engaging (C:3, I:6). Many students found the interface appealing (C:4, I:6) and intuitive (C9, I10, I12). Finally, many students [I7, I9, I10, I11, I12, I13] felt confident about the competencies they acquired playing the game. Students did not raise critiques if not for the brevity of the experience (C:2, I:4).

8.6 Discussion

We wanted to assess the efficacy of Albot Einstein for learning the pH concept and exploring the potentialities introduced by the multimodality in the interaction.

The difference in the efficacy of Albot w.r.t. interactive web platforms is not statistically significant. Also, the effectiveness of the interaction is comparable, as shown by the same content error rate in the two conditions. Though, users spend much more time, on average 50% more, interacting with the chatbot than with the corresponding web interface.

Despite this difference that might indicate a more elevated cost of interaction, and thus a worse experience, UES-SF scores highlight that users in the C condition did not report a lower engagement than those in the control group (RQ2). Further study are necessary to understand if this additional time is a limit of the conversational agent or, as suggested by Spanjers et al. [348], this could lead to a higher substantive engagement. Such a consequence would be proven to increase students' motivation, and consequently to higher long-term performances [349, 350]. In the future we aim at carrying out a new study in which we compare also the information retention in the two conditions.

From a qualitative perspective, users recognize the efficacy of the interaction with the conversational agent, as the majority of them reported during the interviews and the increasing interest in PCA shows [351]. Children are attracted by the PCA and are curious about it. As the interaction continues, curiosity evolves into engagement: the chatbot is perceived as a companion that makes the experience intuitive [C10], as a provider of helpful explanation [C2], and as an agent that can make "fun even subjects that [children] find boring" [C3].

The interaction analysis shows that the design of the chatbot directly impacts users' perception during experience (RQ3), as highlighted by previous studies [342, 351]. As shown by the difference in the number of messages exchanged in the two tasks, children interact much with the chatbot at the beginning of the experience to focus on the GUI in the second part of the activity. A dual nature of the role of the conversational agent in the experience emerges; in the first part, it is a tutor that introduces the children to the activity, guiding them in the interaction while introducing the fundamental concepts of the subject. Then, as children learn how to play, the focus of the interaction is moved to the GUI. Children do not need to be guided anymore but want to act autonomously; a conversation that guides them step by step would be redundant and burdening. At any time, children could ask for further information from the PCA as if it were a teacher for the user. At the same time, in the advanced phases of the game, the PCA can act as a facilitator, providing hints on what the player is doing or increasing the engagement by giving children further challenge to solve at a difficulty level that is adaptive to children's capabilities.

Coherently with existing literature, evidence of the importance of multimodality was also given from how errors occurred and were recovered [352, 353]. 10 participants out of 14 who used the chatbot asked the facilitator which modality to answer the PCA's questions (many were answerable using both modalities). At the same time, 5 participants in the C condition used the chat instead of occasionally clicking on the corresponding elements. These observations show how students feel natural the interaction with the multimodal Pedagogical Conversational Agents since the first step of the interaction. Children immediately understand their capabilities and their usefulness for their learning experience.

8.7 Conclusions

We wanted to explore the adoption of multimodal Pedagogical Conversational Agents for STEM education. To do that, we interviewed four experts who helped us elicit a set of principles for designing tools that can guide the design process.

Following the principles, we implemented Albot Einstein, a multimodal Pedagogical Conversational Agent, to teach children pH concepts. Interacting with it, children can experiment in the first person on how to use red cabbage solution to measure the pH level of a substance, grasping the concepts of acid and alkaline solutions and pH indicators.

We measured the efficacy of Albot Einstein in an experimental study in which we compared the interaction and performances of students that played with Albot Einstein with respect to the ones who tried the application deprived of the chatbot. Empirical measures show no statistically significant difference in learning effect and user engagement in the two interaction conditions. The interviews showed that most of the study participants appreciate the presence of the chat in the experience, recognizing it as the best component of the game-play.

We showed how such a Pedagogical Conversational Agents plays different roles during the interaction experience [354]. The PCA is the tutor that introduces users to the platform to explore the disciplines, supporting users in formulating their scientific hypotheses. It proactively stimulates the child to discover new concepts and refine their observations when needed. Finally, it acts as the teacher that explains the concept behind those observations, ensuring the correct understanding of the underlying scientific principles. Roles are not constant, but the chatbot can switch from one to another, adapting to the user's particular needs and ensuring the best experience possible.

Also, the locus of the interaction changes while the experience is evolving; our observations show how users tend to rely more on the chatbot in the first phases when they have more need to be guided, to then concentrate more on the GUI once they understand how the application works.

Our research is not exempt from limitations. Despite being rigorous, our study only concentrates on the immediate retention of the information rather than investigating the long-term. In the future, we aim to test our product on a broader audience and verify its effectiveness in the long-term scenario. In addition, we want to test such agents in others subject to see how they perform in various domains.

Our research explores the study of multimodal Pedagogical Conversational Interfaces that experience designers and educators can leverage to stimulate the benefits of informal learning in a digital environment and create games that can raise children's awareness and interest in STEM disciplines.

However, the design process of Albot Einstein required a custom technological solution that required a considerable time investment for its realization. In the next chapter, we present an online authoring tool that allows the generation of a backend that supports multimodal conversational interaction through a drag-and-drop block interface.

Chapter 9
A Low-code Authoring Tool for Multimodal Conversational Agents

9.1 Introduction

In the previous chapters, we defined a model to describe strongly integrated conversational agents and showed an example of how to employ it to design Albot Einstein, a multimodal pedagogical conversational agent. Although the use of the model was successful, the model we presented supported Albot Einstein's realization only during the design phases, requiring us to create a custom architecture to manage multimodality during conversational interaction.

We want to automatize the creation of the structure of the conversational agent. We observe that multimodal task-oriented conversational agents, even if support different processes, present common functionalities in the backbone structure: a process manager that rules the interaction, a conversational engine that translates users' input into natural language and produces conversational outputs, and a system to manage inputs and outputs on the others modalities.

In recent years, low-code and no-code programming paradigms emerged to enable a broader population to ease access to programming software. These are platforms on which users can typically define the behavior of the programs through graphical user interfaces, and the platform automatically transforms them into executable software [355].

The difference between the two families is thin and related to the possibility of customizing the behavior of the programs by adding user-written code, whereas in no-code applications users only can compose pre-defined blocks [356].

Today, there are low-code/no-code applications for many domains, such as web development, smartphone application programming, coding education, robotics, and many more. The recent success of conversational agents leads to the birth of such chatbot platforms. Conversation designers can define the conversations typically through some flow diagrams or question-answer pairs, provide examples of users' utterances to train the natural language processing engine, and the platform automatically deploys a conversational agent ready to be inserted on an existing interface or used through common messaging applications.

However, Multimodal Conversational Agents do not profit from a generation tool designed with this form of interaction in mind, despite the recent breakthroughs of this new paradigm and the popularity of chatbot-generating tools.

For this reason, we want to build on top of our model a tool that allows programmers to define a multimodal interaction flow graphically, and it automatically carries out several backend setup tasks, such as communication structure, event handling, and NLP engine maintenance.

In this chapter, I will describe the design and the implementation of this tool, bringing major contributions:

- An analysis of the existing conversational agents authoring tools;
- The design and the implementation of an interface to author multimodal conversational interfaces;
- An empirical evaluation of the interface.

9.2 Background

9.2.1 End-User Development

End-User Development (EUD) refers to tools that let non-professional developers build complex data objects and software products without having to have a deep understanding of programming [357]. Scripting languages, spreadsheets, graphic programming, trigger-action programming, and natural language programming are just a few instances of programming by example.

The most utilized EUD tool is a spreadsheet [358]. They are programming environments with first-order functional programs as their formulas [359]. The outcomes of the formulas are the calculated output values, and the said formulas may relate to input variables denoting cell names.

The programmable environment is portrayed in end-user languages as visual metaphors that adhere to condition-action principles. The goal is to lessen the cognitive load by closing the conceptual gap between programming and real-world operations [360]. For instance, while using a spreadsheet, entering a value into a particular cell corresponds to initializing a variable with that value and the cell's name as the variable.

9.2.2 Low-code and no-code development platforms

Two new, disruptive paradigms —low-code and no-code development— have been attracting more and more attention among the branches of EUD (figure) [361]. They offer a graphical programming environment that enables people with no programming skills to utilize it and enables programmers to quickly and automatically develop programs and applications.

Low-code platforms enable developers to customize an application for a particular use case or functionality, generally using graphs and forms [361]. These applications are developed utilizing declarative, high-level, and graphical abstractions, cloud computing infrastructures, automatic code generation, and model-driven engineering methodologies [355]. These platforms are primarily cloud-native.

Platforms of this kind are designed for usage by expert developers because low-code programming requires some level of coding proficiency or aptitude for technical subjects. In most circumstances, more granularity and experience ensure that the outcome of this process is of higher quality and more production-ready.

Low-code authoring tools bring several benefits to the programming process, such as saving time and improving overall productivity, reducing the need for specialized skills, and offering a satisfactory degree of freedom to the developer at the cost of some technical skills required.

No-code development, instead, overcomes that barrier, providing easy access to programming only exploiting a graphical user interface at the cost of less flexibility of the tool. IFTTT is the most famous example of a no-code platform. It is an online application that allows the definition of ECA rules (event-condition-action) through the formula *If This Than That*. GeCoAgent and DSBot, presented in chapters 2 and 3, can also be considered no-code platforms.

9.3 Conversational Agent Generation Tools Analysis

This section examines the major players in the panorama of low-code and no-code tools to generate conversational agents. Being those mainly commercial products, we ran our analysis looking at the survey presented in the major technology magazines and blogs. Even if it is not a comprehensive list, we were interested in the technological and design solutions implemented by the most popular and the fastest-rising platforms as case studies of successful paradigms to define conversational interaction.

We analyze the platform along four dimensions:

1. **Input processing.** We examine how the platform processes users' input, whether the processing is transparent to developers, or they must configure a Natural Language Processing (NLP) engine, and which information those engines can identify and extract.

2. **Dialogue Definition.** We examine how the dialogue is modeled and how developers have to describe the structure of the conversation to generate the conversational agent.

3. **Integrations.** We examine how different CA generation platforms may offer different functionalities in their development kits, such as the integration with chat services like Slack, Facebook Messenger, and Whatsapp, or with development-specific

services such database providers, testing platforms, cloud functionalities, or version control software, in addition to functionality service support.

4. **Deployment Modality.** We examine how, once the conversation has been designed, the developer can deploy the conversational agent in an application.

9.3.1 Dialogflow

Google's Dialogflow is one of the most well-known and acclaimed CA-generating tools. It is an entire set of tools that includes everything from the original design of a CA to validation, testing, and user behavior analysis.

Usage Target users for GUI-based applications include not just developers but also non-programmers such as CA designers. However, unlike typical chatbot developers, its user interface is not built on a drag-and-drop procedure. First and foremost, CA designers must initialize *intents* and *entities*. The former specifies all the actions an end user wishes to accomplish, while the latter are the objects of the context of the entire conversation flow.

The potential to develop a conversational agent utilizing pre-built models likewise deserves consideration. By doing so, CA designers may quickly build their chatbot with minimal customization and without the hassle of declaring intents and entities, saving time over starting from scratch.

Integration. Dialogflow has a large selection of plugins for integrating external services. There are all the most popular messaging platforms (such as Telegram, Facebook Messenger, and Slack) and direct telephonic integration, enabling users to communicate with the chatbot by voice.

Input Processing. Dialogflow allows CA designers to train entity/intent recognition using regular expressions. The platform also includes NLP for phrase matching with the ability to extract parameters from the processed text. The platform's enterprise edition allows CA designers to use sentiment analysis to categorize input. Additionally, more than 50 languages support these natural language processing functions.

9.3.2 IBM Watson

Due to its extensive market presence and functional capabilities, IBM Watson[1] might be compared to DialogFlow.

Usage. The IBM Watson user interface provides a full range of tools while maintaining a simple design for new developers. It permits the definition of entities and intentions, just like DialogFlow. Machine learning enters the picture when developers describe such variables; thus, CA designers are urged to use as many instances as possible. By relying on

[1] `https://www.ibm.com/watson`

the definition of blocks, it is also feasible to structure the dialogue into a tree-like structure. Each block represents a phase of the chatbot dialogue. CA designers can match the previously defined entities and intents inside each block.

Integration. Dialogflow presents many integrations for deploying conversational agents on multiple channels (chat applications, text messages, phone calls, etc.) and supporting popular applications, such as web searches, CRM applications, and databases.

Input Processing. A powerful NLU engine is a critical component of IBM Watson. Speech recognition, phrase parameter extraction, and text processing are all fully supported. When raw text is provided as input, sentiment analysis—which is also supported—helps identify the most appropriate entity and intent.

9.3.3 Amazon Lex

Lex is a platform for natural language processing offered by Amazon Web Services (AWS). Amazon provides the essential characteristics necessary to create a pre-built conversational agent in addition to the NLP engine.

Usage. The user interface for Amazon Lex does not use conversational blocks. The CA developers must set up all the actions they anticipate chatbot users to take because the interface is *intent-based*. Setting up intentions also entails giving several examples of questions: Automatic processing by the NLU engine results in updates to the conversation agent model. Additionally, the platform enables developers to create entities, or *slots*, which CA designers may then put in persistent variables.

Integration. Although having coding skills is not a requirement, Amazon Lex functionalities can be further expanded using the AWS domain because CA developers can interface their conversational agent with every Amazon cloud computing tool accessible. Software developers can increase the functionality of their agents by having them run custom code using AWS Lambda. Like DialogFlow, development-specific features are restricted to the Amazon environment, and integrating cloud services other than AWS is challenging.

Input Processing. Lex's robustness is dependent on Amazon Web Services (AWS). Amazon guarantees to have all the machine learning resources necessary to maintain a powerful NLU engine. Lex does provide NLP support for parameter extraction and storage (the slots), ranging from basic regular expression to in-depth sentiment analysis.

9.3.4 Azure Bot Service

Azure Bot Service is the set of conversational services provided by the Microsoft Azure computing platform.

[1] `https://aws.amazon.com/it/lex/`

[1] `https://azure.microsoft.com/en-us/products/bot-services/`

Usage. Azure Bot Service requires the knowledge of C# programming language and programming skills to set up the project, integrating the various services offered by the platform.

Integration. The Azure Bot framework provides native channel integration. The intermediate Bot Framework tier receives messages from several supported sources and integrates and transmits them to the Bot program, which forwards them to the NLU engine. Additionally, advanced developers can create their own external channel integration with moderate effort.

Input Processing. After the code setup is finished, CA designers can use the LUIS natural language processing engine. This Microsoft NLU utility offers all the standard functions for processing input: Speech recognition, regular expressions, NLP, parameter extraction, and sentiment analysis.

9.3.5 FlowXO

FLowXO is an online service to create a chatbot centered around the idea of *conversation flows*.

Usage. This service creates a flow throughout the entire chat. A linear list of blocks is used to provide the user interface. By placing conversational chunks along the stream one below the other, CA designers can construct the chatbot experience. A piece of logic known as a *conversational block* allows CA designers to choose the entered answers and offer responses. Simple text responses, selections, and yes/no dialogue confirmations can all be sent.

Integration. In order to connect with social networks, FlowXO offers adapters. In addition, a chatbot can call outside services to improve functionality. User activity analytics are also supported.

Input Processing. Native natural language processing is not supported; developers are requested to enter many potential user responses or utilize regular expressions. Additionally, filters are used to produce conversational logic. CA designers can project conversational paths by enhancing end-user historical responses. There is no branch visualization because the entire discussion is structured along a single linear flow, including optional utterances. For these reasons, FlowXO is better suitable for designing straightforward chatbots or task automation executors than for producing conversational agents that resemble people. Its most significant advantage is simplicity: a CA creator without programming knowledge may perform an entire automator in minutes.

[1]`https://flowxo.com/`

9.3.6 RASA

RASA is an open-source framework to create task-oriented conversational agents.

Usage. Most of this framework's experience is code-free, and CA designers can only construct conversational agents using the GUI. Despite this, RASA provides a complete development kit to expand its features. Using the library offered, it is possible to create customized agents that can do operations programmatically, retrieve data from external services, or carry out operative system tasks. The RASA UI is jam-packed with features; therefore, this framework has a steep learning curve. However, beginners can begin developing their own CA by selecting the *interactive learning* option, which enables training the agent through interactive chat using real-world examples.

Integration. RASA provides many options for integrating outside services. CA designers have access to the library or Twilio integration tools. If this is still insufficient, the integrations of RASA can be expanded by forking the project.

Input Processing. The primary purpose of RASA is to process natural language using neural techniques. Anyway, regular expressions can be used to identify basic speech patterns without any training quickly. Moreover, RASA facilitates the extraction of parameters from phrases in order to construct new entities and intents. Both volatile and persistent parameters are possible.

9.3.7 Xenioo

A comprehensive chatbot generation platform is available from Xenioo. It is thought for CA designers without any programming experience.

Usage. Making a personalized conversational agent is a simple process. Different *behaviors* that correlate to various end-user intents can be created by developers. Each behavior is linked to a distinct canvas where a graph of interactions can be defined. Each interaction is tied to the questions the conversational agent asks the user. Depending on the user's responses, the flow moves to a different intended interaction. Multiple interactions and multiple ones can address a single interaction. The resulting structure is a graph, and interactions are therefore represented by blocks that are logically connected by links. Additionally, CA designers can start with pre-made conversational agent templates and modify them to suit their requirements.

Integration. In order to install the chatbots on the most popular channels, Xenioo provides a number of ready-made plugins. Additionally, programmers can design their social network connectors.

[1]`https://rasa.com/`

[1]`https://www.xenioo.com/`

Input Processing. Numerous input recognition features powered by machine learning are available on Xenioo. Using regular expressions, extracting sentence parameters, and NLP are all supported. Xenioo provides a few functional variables that can be used in the conversation flow to the CA designer, and the latter can also create new ones based on prodded utterances. The platform supports multimodal chatbot input. CA designers can make non-textual interaction blocks. There are several acceptable types of responses, including buttons, targeted input listening (perhaps with NLU), and option selections. However, developers can only add input elements within chat and customize them using conventional technologies; therefore, even if this multimodality support is quite robust, it is not a complete integration.

9.3.8 Landbot.io

Landbot.io is a no-code conversational agent generation platform.

Usage. Landbot.io is based on a straightforward graphical user interface, and CA designers can create their conversational agents by using a canvas with pre-built blocks. By dragging and dropping the required blocks and connecting them, it is possible to build a conversational flow. Each block corresponds to a prompt question that will be presented to the user—various methods of requesting feedback, such as multiple choice with buttons.

Integration. Landbot.io can be natively integrated with web applications and Whatsapp and provides a set of pre-built integrations with CRMs, CMSs, and third-party applications.

Input Processing. The creators of conversational bots must rely on regular expression matching because this platform, like FlowXO, does not allow natural language processing. This platform is better suited for the straightforward development of conversational agents with minimal developer experience.

On the other hand, DialogFlow integration was made possible by Landbot.io. The canvas can then import a DialogFlow block to take advantage of its NLU features. By doing this, CA designers can preserve a simple, graph-based dialogue representation while relying on a reliable natural language processing engine. Landbot.io also offers a rudimentary multimodal chat capability, similar to Xenioo: instead of sending the user only text, CA designers can send choice buttons, forcing the user to respond unequivocally.

9.3.9 Chatterbot

Chatterbot is an example of a programming library, specifically in Python, that makes it simple for programmers to create conversational agents and handle user input. In contrast

[1]`https://www.landbot.io`

to the software produced by platforms or frameworks described before, this library's output is still being prepared for deployment since it must be inserted in a web application. On the other hand, developers are granted more authority, and service integrations have no restrictions.

Usage. Being a Python library, CA designers need to be able to code with Chatterbot. Developers must include optional integrations while initializing the chatbot tool and importing the constructors (Database URL, storage adapter, and logical adapter). The chatbot instance can then be "trained" to recognize input data: Chatbot creates conversational graphs that represent the interactions' cycles. Developers are not required to explicitly set up loops or branches because logic is performed internally by providing conversational input, which is then turned into the graph.

Integration. Being a code library, Chatterbot requires developers to integrate it with every service they want manually. Natively, this library only offers integration with the Django framework to build the server hosting the conversational agent.

Input Processing. By focusing on the generation of graphs, Chatterbot provides a native natural language processing engine. However, this functionality is constrained because sentiment analysis and other advanced language processing features that rely on machine learning are not supported. However, developers can integrate external NLU services (like LUIS or RASA) that offer APIs by creating their custom adapters.

9.3.10 Pandorabots

Pandorabots is a framework for creating conversational agents targeting conversation designers and developers without coding knowledge.

Usage. Developers can use various techniques to build their agents using a dashboard view. CA designers can rely on an interactive conversation design modality where user intents can be defined. On the other hand, using Artificial Intelligence Markup Language can result in more control (AIML). With that information, programmers created a processing category that has two properties: pattern, which denotes the user input that needs to be intercepted, and template, which denotes the relative chatbot utterance.

Integration. Some external channels have native integration in Pandorabots. Custom interaction support can be developed for particular social networks.

Input Processing. Developers must match and record all potential patterns resulting from user input using AIML. This recognition is not based on a machine learning engine

[1]`https://chatterbot.readthedocs.io/`
[1]`https://home.pandorabots.com`

but on regular expression matching. Therefore, this strategy can be viewed as being extremely inadequate, especially when dealing with unanticipated user input.

9.3.11 Xatkit

Xatkit is the result of an alliance between Berger-Levrault and the Open University of Catalonia [362]. It was created to simplify the deployment of chatbot services in government portals, to be tailored around the e-commerce experience.

Usage. Xatkit includes a robust software development kit (SDK). Programming in a domain-specific language (DSL) is the development methodology. CA developers must write the code for their Xatkit conversational agent in the platform's programming language before compiling it. Options exist for recognizing intents, entities, and extensive data.

Integration. Xatkit can be easily integrated with WooCommerce and WordPress to enhance the customers' shopping experience.

Input Processing. Xatkit presents several built-in conversations that the developer can expand by adding FAQs question-answers pairs from a dedicated GUI.

9.3.12 Discussion

The conversational agents generating tools discussed above come in various varieties. The most common method of tool provision is through an internet portal (PaaS). Others are provided as frameworks, allowing for on-site installation. The only service offered among those listed is Chatterbot, while others allow use as external callable services. These variations increase the accessible feature set and complicate comparability.

However, from a functional standpoint, most of them rely on the paradigm of conversational creation. It is necessary to determine the most likely explanation in a more straightforward scenario. Most of the time, this results in a close relationship between the conversational agent's actions (the intents) and how these actions are represented (the end user's utterances), which may cause observable development bottlenecks.

Multimodality is only offered in the context of chat when it is accessible. Because controls on website UI elements are never provided, this limits the options available to CA developers. Many CA frameworks do not provide any multimodality support at all.

The construction of complicated conversational bots may be severely hampered by the fact that NLU-related properties are not always available. Additionally, most of the time, AI-powered applications are proprietary, making integrating them with other services challenging. On the other hand, the modular design of RASA and its ability to be accessed through an external API help to address the drawbacks mentioned above partially.

[1]`https://xatkit.com/`

Nearly all of the tools offered for dialogue design mode rely on a GUI. The GUI often makes it simple to define intent and entities, and it occasionally uses a block-based interface to describe the conversational flow. In RASA, hybrid development—which uses code and a GUI—is only available as an advanced capability.

These considerations are the foundation of the platform design we will present in the rest of the chapter.

9.4 System Overview

Our platform is an online authoring tool that allows to:

1. Design the interaction flow with task-oriented multimodal conversational agents;
2. Define the events on every modality that make the interaction proceed;
3. Define the system feedback on every modality;
4. Specify advanced custom operations through snippets of code;
5. Train a conversational agent to recognize users' utterances;
6. Deploy and manage running applications, exposing a set of APIs developers can use to integrate the multimodal frontend.

Since, coherently with the model described in Chapter 7, the whole representation is centered around the concept of *event*, this authoring tool is agnostic from the modality: it can work with the combination of a task-oriented conversational agent (written or text-based) and any other modality (one or more). The frontend will be in charge of translating the events into outputs declined on the various modalities.

Once logged in, the developer can select the project on which to work from a list of all the available ones. Once a project is opened, developers can define the Process Diagram on a canvas by dragging and dropping task blocks and gateways.

Then, they can open every item in the Process Diagram to define its Interaction Diagram. The authoring tool supports a condensed version of this diagram where the state machines of all the modalities are condensed into a single one. In the Interaction Diagrams, developers must specify which are the output events, i.e., that will be processed by the frontend to generate the (multimodal) output, the snippets of executable code necessary to fulfill the task (e.g., third-party services calls), and the users' events that correspond to the transactions on the Interaction state machines. If these events must be recognized from user input, developers must provide sentence examples so that the Natural Language Unit can be trained accordingly.

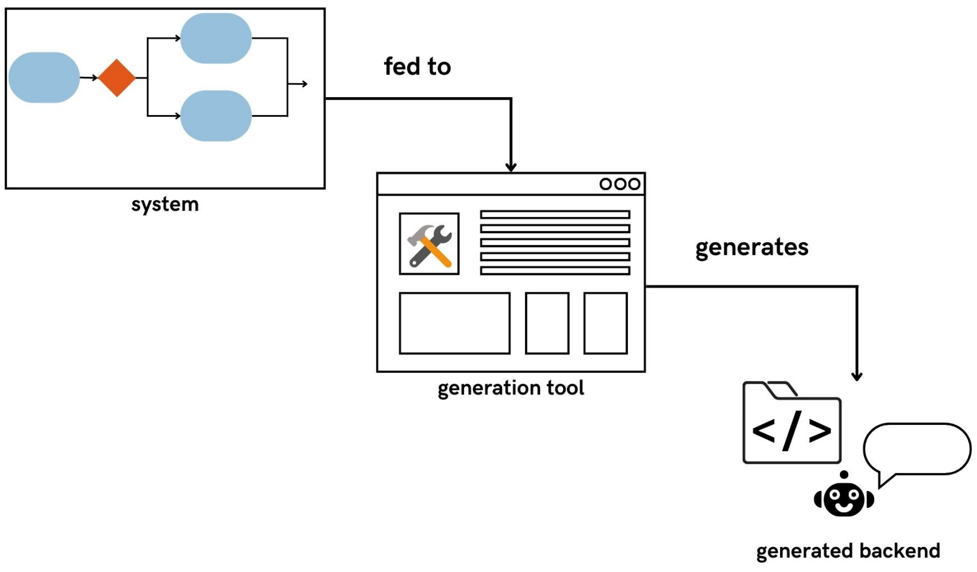


Figure 9.1 – *An overview of how the system works.*

Finally, with only the push of a button, the system automatically transforms the project into an executable backend, comprising the parsing of users' utterances and the management of the state machine, and provides developers with an API-based interface to integrate it with the frontend. Figure 9.1 shows a high-level representation of the generation process.

9.5 Implementation

9.5.1 Frontend

The frontend is a web application developed in React. It is composed of five main pages:

1. A login page, where developers can authenticate;
2. A projects page, where developers manage their projects;
3. A process diagram authoring page, where developers define projects at task diagram level;
4. A task diagram authoring page, where developers define projects at interaction diagram level;
5. An event manager page, where developers define user events and provide samples to train the NLU unit.

Once developers log in, the platform shows them the projects they created. They can create a new project or select an existing one from the projects page.

When a project is selected, the process diagram authoring page opens. The process definition occurs through drag and drop interaction: users can select the type of blocks they want to insert from the list on the left and drag them on the canvas. Then, they can connect the blocks to define the order of the tasks. The closing symbol is automatically added when a gateway is added to the project. An arbitrary identifier is given to the blocks added; developers can modify them to increase the readability of the diagram.

Developers can access the interaction diagram of every task and gateway through the options menu associated with each block. The interaction diagram page interface is similar to the process diagram one, except for the type of blocks users can add to the process.

Every block in the diagram presents a drop-down menu from which the developer can specify the event linked to the transaction that brings to the block itself. All the events already inserted in the project are present in the list. Developers can click on the "Add new event" option to open the event manager pane. When a new event is added, the developer must specify if the event is generated through a message in natural language sent by the user or if it is generated by the other modalities or the system. In the first case, developers are also required to insert a set of training samples for the NLU unit.

Finally, once developers define the whole diagram, they can deploy it by pushing the play button. At that point, the whole project is transformed into a single JSON representing the schema, which is validated through a formal algorithm that checks if the schema is valid. If so, it is merged with the information about the events and sent to the backend so the deployment can start.

9.5.2 Backend

The backend is a server in Python that, in addition to communicating with the frontend and providing it with the information it needs to run the application, is responsible for transforming projects into executable instances and managing them.

The JSON received by the frontend is split into its parts and sent to three main modules:

1. The state machine generator, in charge of generating the data structure representing the state machine on which the interaction will be based at runtime.
2. The event trigger generator, in charge of configuring the event trigger module, i.e., the unit that receives users input from the backend and triggers the corresponding event that makes the state machine proceed.
3. The NLP generator, in charge of creating and training a RASA instance for the running project.

The generated project comprises a folder whose structure is illustrated in Figure 9.2. Inside the project folder, we find a handler for events triggered by the frontend and generates the responses, one that manages the connection with the user application through Python WebSocket, a file that manages the state machine, and a folder containing the NLU module, with the handler and the model trained during the generation.

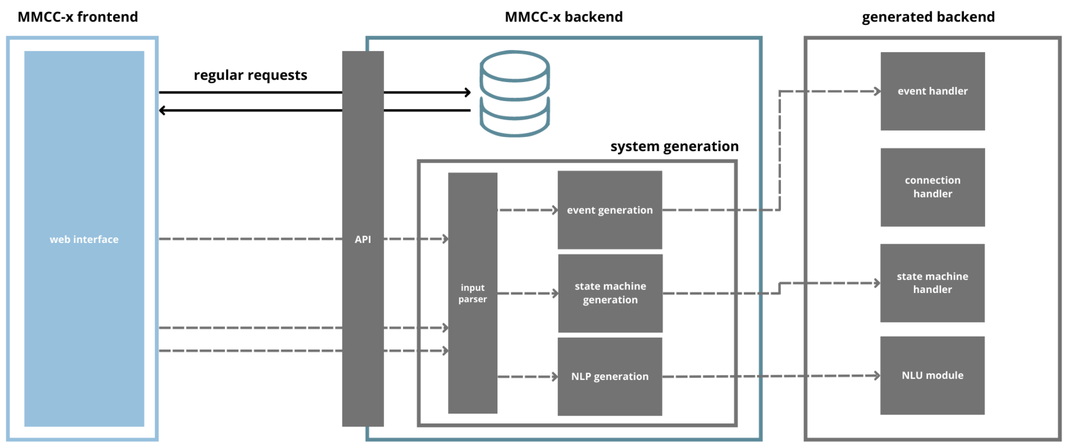


FIGURE 9.2 – *Backend genreated*

9.6 User Evaluation

We conduct an empirical evaluation with 15 participants to assess the efficacy and usability of the authoring tool. During the study, we gather quantitative and qualitative variables to answer the following questions:

- **RQ1:** is the tool perceived as usable?
- **RQ2:** are developers able to use it in autonomy to define the backend structure of a multimodal conversational agent?
- **RQ3:** which is the potential for adoptability of the tool?

9.6.1 Participants

We recruited 15 participants on a voluntary basis. To join the study, participants had to have a degree in computer science or work as a developer for at least three years.

None of the users had seen or used the tool before, and their average ages ranged from 23 to 30 (M = 25.27). Participants in this study took on the role of Conversational Agent designers and identified them through a code (P-1 to P-15). Before the study, the participant gave their agreement by signing a paper describing the objectives, procedures, and legal standards followed to ensure the confidentiality and anonymity of the data collected.

9.6.2 Setting

The study took place in a dedicated room in presence. The participant sat at a table in front of a computer, with a facilitator next to her/him, whose goal was to illustrate the task and guide the interview. On the side of the table, an observer looked at what the user was doing through a screen that mirrored the participants' one and took notes about the interaction.

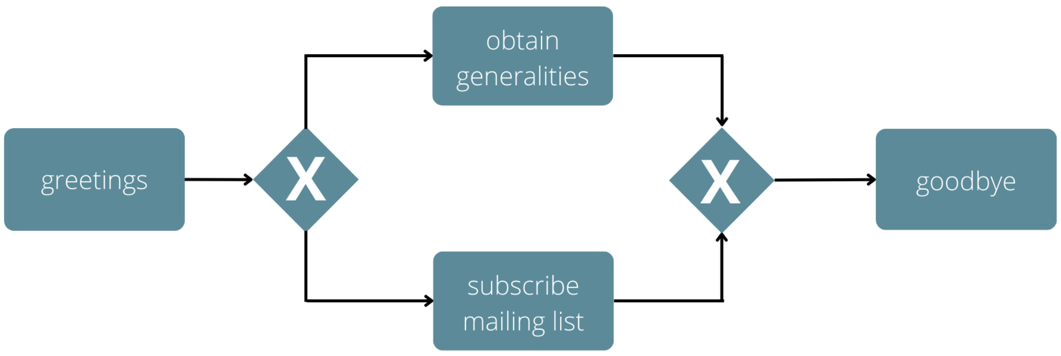


Figure 9.3 – *Process diagram of the conversational agents that study participants had to implement*

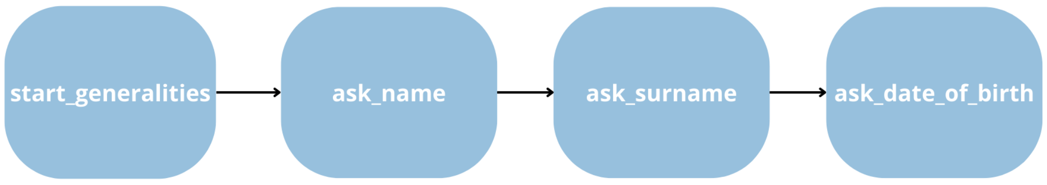


Figure 9.4 – *Interaction diagram of the "obtain generalities" task that study participants had to implement*

9.6.3 Procedure

The experimentation consisted of individual interviews, was divided into four phases, and lasted around 25 minutes per participant.

Phase 1: Demographics and introduction.

Users were welcomed by the facilitator, who asked them to fill out an anonymous form about their demographics. We collected age, gender, education level, and degree of familiarity with chatbot programming. Then, participants were introduced to the interface and the conversational agents they would have to implement during the session, represented in the diagram in Figure 9.3. We gave them the description in the form of the task diagram since the design of the interface through the model was not part of the evaluation.

Phase 2: Project creation and process diagram definition.

Participants had to create a new project from the project page, open the process diagram authoring page and implement the diagram shown in the previous phase (Fig. 9.3). We asked participants to *think-aloud* in order for us to be able to understand the reasoning users were making when using the interface.

Phase 3: Interaction diagram and events definition.

Once defined the process diagram, users had to open the interaction diagram page corresponding to Obtain Generalities task, illustrated in Figure 9.4. In addition, they had to define all the events included in the diagram. Participants were free to decide on which modality the event would have been triggered, with the constraint of defining at least two events launched through textual messages and providing at least two examples to train the NLU unit.

Phase 4: Final Evluation

In the last part of the session, we asked participants to answer two evaluation questionnaires: SUS [180] and NASA-TLX [182]. Finally, we conducted a semi-structured interview based on the following questions:

1. Which elements of the application did you like the most?
2. Which elements did you like the least?
3. Would you use the tool in a real project?
4. What would you modify inside the interface?

9.6.4 Results and Discussion

The user evaluations of usability that have been gathered, along with the remarks that users have provided in the part of open-ended questions, indicate a very positive impression. According to Sauro [123], the tool's score on the System Usability Scale surveys, using the Brooke-identified method [180], was 83.3 with a deviation of 7.8, placing it in the top 10% and designating it as Excellent.

According to Grier [186], using the findings of the NASA TLX to assess the workload of the tool on the user, we received an overall score of 40.08, which puts the product barely above the 30% percentile. This score, along with 50% of the measured global workloads, lies between 36.77 and 60.00.

With a standard deviation of 1.52 and 2.01, 100% of users could complete both tasks in an average of 4.35 minutes for the first task and 7.29 minutes for the second.

These assessments were examined using a thematic analysis [119] and went well with the user's general remarks in the last unstructured interview. The tool was deemed simple to use and learn by eight users [P-2, P-3, P-4, P-8, P-9, P-10, P-11, and P-13], while it was deemed intuitive by two users [P-8, P-13], as well as simple to use [P-6], efficient [P-10], and accessible [P-11]. Four users spoke about how much they appreciated the tool's clarity, which was communicated through its graphical aspects, and how *"blocks and connectors considerably aided their thinking process,"* [P-2, P-4, P-11, P-13, P-14] in a user's words [P-9]. The same user clarified that they believed the tool's simplicity resulted from the fact that most of the principles it uses, such as block schemas and gateways, are well-known to computer scientists.

Users who frequently go through the process of creating an MCA as part of their job or research [P-4, P-5] and students or anyone looking to approach chatbots [P-1, P-2, P-8, P-9] are listed as the tool's primary stakeholders and beneficiaries, but in reality, anyone with instructions [P-3, P-7] can use it.

Additionally, all of them said that they would use this tool if they were to construct a multimodal chatbot. This tool was specifically viewed as appropriate for straightforward projects [P-1, P-2, P-4] that do not go to the production stage [P-4], for educational or academic objectives [P-4], and generally as a substitute when it is not necessary to create the agent from scratch [P-5].

Users contrasted the tool with other graphical tools that are currently available, like draw.io (now diagrams.net[2])[P-4, P-14], tripetto[3] [P-12], and Dialogflow [P-10, P-15]. Most comparisons were used to showcase this tool's advantages and draw attention to its functional shortcomings in contrast to other tools. Despite the contrary, users still thought this interface should offer comparable commands to interact with the canvas as the other graphical tools. Their justification for this is that consumers become frustrated when expected behaviors associated with actions like a double-click or right-click do not materialize [P-4].

Users talked about the tool's usability, and clarity of the model used when using it or commented on specific aspects.

The model's hierarchical structure was well-liked by users [P-6, P-13] and, in the words of P-6, *"makes things easier for larger models."*

The fact that events of a particular transaction were to be associated with the condition they caused rather than the transaction itself was the trait that sparked the most debate and confusion. Almost all users were confused about where to add the events [P-2, P-3, P-5, P-6, P-7, P-8, P-11, P-12, P-14] as a result of this unnecessary complication. As a result, many users assumed the event was generated by the backend rather than the one they had received when they noticed it graphically inside the state. Direct suggestions were made to move the event from the state to the transaction by P-2, P-4, P5, and P-6.

Users frequently pointed out notation as a problem as well. Three users [P-4, P-5, and P-10] needed the assistance of facilitators because they struggled to decide whether to use a state or a logic block. A similar issue occurred when defining events: only five users correctly identified the events to be used [P-4, P-5, P-6, P-7, P-8], and six users were unable to fill out the utterances field [P-1, P-3, P-7, P-11, P-12, P-13] when given the option to choose whether an event is ui-related or conversation-related.

The interface offered no explanations or guidance in either instance, but it stands to reason that notations issues would diminish following good onboarding and with instructions or a guide attached to the tool.

[2]`https://app.diagrams.net/`
[3]`https://tripetto.app/`

Many errors and comments also came from the Start and End blocks. Almost all users instinctively added them to the model when they saw them in the menu. However, because they thought of them as tasks, they sought to change their names [P-5, P-6, P-7, P-8, P-10] or used them for “welcome” and “goodbye” activities [P-1, P-2, P-5, P-6, P-7, P-8, P-10, P-13, P-14]. P-13 suggested allowing users to mark jobs as initial or final while not separating them from typical tasks. P-11, on the other hand, recommended that since they are necessary, they should be fixed someplace on the canvas from the start.

Two people also provided feedback on the magnitude of the events. P-6 and P-15 initially believed that events were local to a state and not global because all 15 users generated events as required to do so and used the “add event” feature that showed up in each state’s drop-down menu. Even though it wasn’t obvious at first, P-6 said that it *“makes sense and is useful that events are genuinely global.”*

Minor problems with the user experience were also found, including broken connectors and an offset in the dropping of objects onto the canvas. However, because they had a minimal impact on the user experience, they were declared irrelevant to the discussion.

9.7 Conclusions

In this chapter, we presented the process that led to the design of an authoring tool to graphically generate the behavioral logic of multimodal conversational agents. We started by analyzing the available platforms for authoring conversational agents to understand the most successful interaction paradigms that are worth exploring.

Then, we implemented an authoring tool based on a drag-and-drop block interface that developers can use to build their application’s backend graphically. The programming process consists of three steps: the definition of the process diagram, the interaction diagram for every process block, and the events that trigger a transaction in the state machine. Then, with the push of a button, the structure is validated, and, in case of a positive outcome, the backend is automatically generated.

An empirical evaluation shows the potential of such an interface: the tool is perceived as easy to use and has a high potential for adoptability. Developers appreciate the intuitiveness of the block-based interface, the possibility of rapid prototyping strongly integrated conversational interfaces, and the possibility of concentrating on the interaction design rather than on the technicalities of the implementation.

Our platform is not exempt from limitations; it must be considered as a prototype that needs to be expanded, introducing additional features and functionalities, such as collaborative options and a more powerful authoring interface. However, it is the first attempt of a new class of tools to create more powerful conversational interfaces that support users at 360-degrees.

Chapter 10
Conclusions

10.1 Discussion: a summary of the contributions of the thesis

We are approaching the end of this journey. We started with designing a multimodal conversational agent for bioinformatics analysis to discover the need to shed light on theories and practices to design applications based on those agents.

In this section, we will briefly revise the contributions presented so far to propose a holistic view and combine them in a unique framework in the next section.

In Chapter 2, we saw how to design a conversational agent on a process ontology. We started from the ontology definition and its validation to build GeCoAGent, a multimodal conversational agent for data science, to support the bioinformatics tertiary analysis process, exploiting a novel design technique based Hierarchical Task Trees [83] and filling the gap highlighted by [31]. GeCoAgent's innovative structure is grammar-based, so new modules can easily be added through the extension of the grammars that rules the conversation. We validated GeCoAgent effectiveness with biologists and bioinformaticians to show how the application can support both user categories, supporting the findings of [126, 128, 129].

In Chapter 3, we presented DSBot, another multimodal conversational agent for data science that evolves on the lessons learned with GeCoAgent. DSBot is agnostic from the data since it accepts any data in tabular form. This application is innovative since exploits a mix of technologies (such as conversational agents, machine translation, and autoML) in a transparent way to translate users' research questions into operative pipelines and execute them [150], supporting the process of data democratization described in [132]. In addition to the application itself, the chapter presents how one of the two modules that make up the application has been turned into an open-source framework for integrating a conversational troubleshooting system into web applications, overcoming some of the most common problems of today conversational troubleshooting systems [170]. Finally, we provide empirical studies to measure the machine learning module's functioning and the interface's usability.

Chapter 4 reflects on the lessons learned from the experience with the two conversational agents to set the basis for the generalization process. After defining the research space and introducing the terminology adopted in the following work, we conceptualize the Multimodality Continuum, which is the specter of possible integration paradigms of conversational agents with other modalities.

In Chapter 6, we started to explore multimodality by analyzing the influence that its introduction brings to users' linguistic behavior, trying to better investigate the principles on which previous multimodal conversational interfaces had been built [18, 45, 257, 260]. Through an empirical study, we discovered that when graphical hints are added to the conversation, users tend to make shorter interactions and fewer errors.

Chapter 5 complements the study by investigating the design perspective of the introduction of multimodality, trying to fill at the intersection between the research on usability for multimodality [125, 230] and the one for design the design of conversational agents [240–242]. Examining existing literature, we extract valuable lessons and cluster them into seven principles that can inform the design of such interfaces.

Chapter 7 condenses all findings into a model for designing multimodal process-oriented conversational interfaces, where the modalities are strongly connected and communicate with each other and the user. The model takes inspiration from BPMN formalism to describe the high-level steps users must accomplish to reach their goals. Then, with similar diagrams, developers and designers can describe the interactions in every modality and how they relate. Our model takes inspiration widely-established formalisms to introduce a paradigm shift from a conversational-driven model to a process-driven one [196, 311, 312].

The chapter ends with a proposed architecture based on a shared context to transform the model into the backbone of an application.

Chapter 8 takes up the case study presented as an example in the model description to present it in detail. The application is Albot Einstein, a pedagogical conversational agent to teach children chemistry, precisely pH. This chapter, in addition to describing the process that led to the design of the application by showing the result of a design process that followed the model, empirically evaluates the effectiveness of the platform in one school, exploring by the increasing interest interest in PCA in the scientific community [351]. The gathered data shows no statistically significant difference in students' performance with respect to the ones that interacted with a similar application but without the chatbot, pointing to Albot as a effective tool for education. The interaction analysis shows that the design of the chatbot directly impacts users' perception during experience, as highlighted by previous studies [342, 351].

Finally, Chapter 9 shows the authoring tool we designed to transform the model-based specification of a conversational agent into an executable application backend. Developers can set the process and the interactions through a drag-and-drop interface and train the conversational engine on a dedicated interface. Then, the application is automatically deployed with the push of a single button. We assessed the platform's usability in an empirical study involving developers and software engineers.

10.2 Results: a holistic view on the thesis contributions

As discussed, this thesis consists of a long journey in which many contributions followed one another, even of a heterogeneous nature, linked by the logic of the an overall vision,

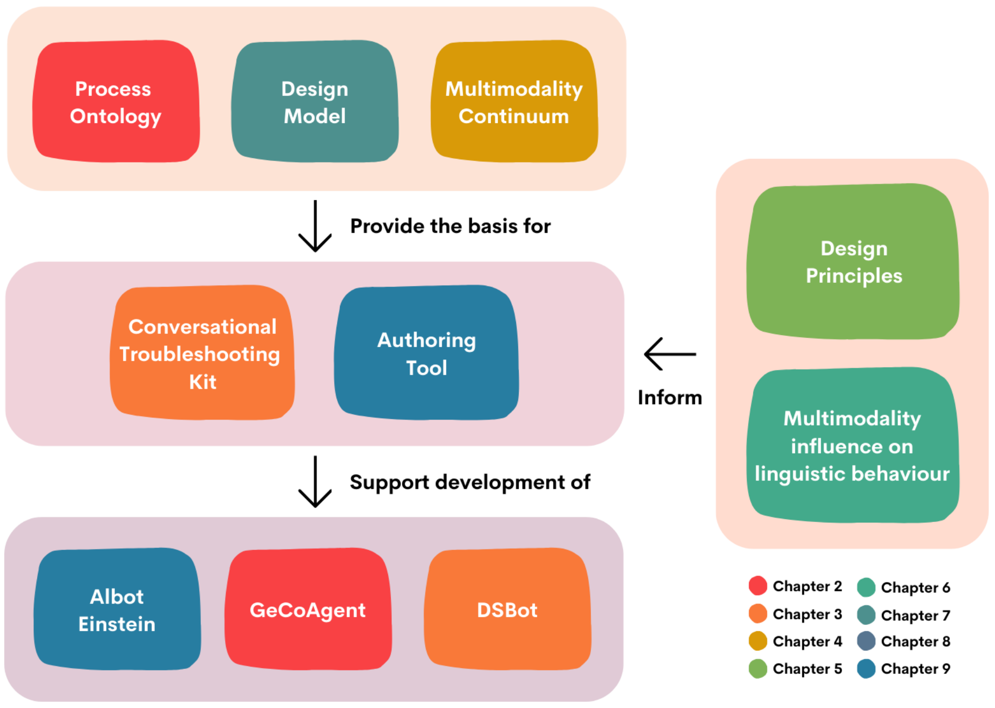


Figure 10.1 – *An holistic view on the contributions of the thesis*

presented in Section 1.2.

In this section, rather than focusing on the single contributions of the thesis, we present the contributions from a different perspective that transcends the temporal narrative on which the thesis has been set so far to give a holistic view in which the various pieces blend in a single vision, as shown in Figure 10.1.

The work described in this thesis can be seen from an holistic perspective that provides small but consistent contributions on all the aspects of the design and implementation process of a multimodal conversational agent.

The ontology presented in Chapter 2 and its elicitation process, the Multimodality Continuum (Chapter 4), and the Design Model in Chapter 7 are methods that provide the basis for the design of a multimodal conversational application. In particular, the analysis that leads to the definition of the process ontology is a methodology that can guide the elicitation of users' requirements in the design of an application. Such a methodology provided a graphical artifact that represents what emerged during the elicitation process, on which all the stakeholders can discuss and agree upon. When requirements are prepared, and the modalities are chosen, the multimodality continuum provides a reference to decide the best degree of integration between the modalities. Finally, the design model allows the researchers to graphically define the whole interaction and works as a reference for designers and developers, minimizing the miscommunications in the development process.

The results of the linguistic study presented in Chapter 6 and the principles elicited in Chapter 5 support and inform the design process. Conversation experts and interaction designers can use these tools to produce a more informed experience design.

The process, so far explored in the design and the methodology, is then analyzed in the development phase through the study of the tools presented in the thesis, particularly the authoring tool described in Chapter 9 and the framework in Chapter 3. The application can be created from scratch or an interface on which conversational capabilities are added. In the first case, developers can use the authoring to quickly transform the design model of an application in a working backend. In the second one, the conversational troubleshooting kit allows them to provide users with conversational support on an existing web application with minimal effort.

Finally, GeCoAgent (Chapter 2), DSBot (Chapter 3), and Albot Einstein (Chapter 8) provide three examples of multimodal conversational agents, developers and designers can take inspiration from. Each presents distinctive design choices that lead to different ways of exploiting multimodality.

10.3 Limitations and Future Works

As discussed in the previous chapters, every study had its limitations due to many factors that directly or indirectly impacted the execution of the work. The main one deals with most user evaluations' small-scale and short-term design. Additional evaluations are necessary to generalize the results. Many works require further testing sessions with more participants, and studies that take two or more sessions in a prolonged time interval to test how the findings apply in an ecological setting [363], i.e., in settings where the tools are used for working in a real setting and with actual tasks.

Another problem we faced was the need for shared terminology in the conversational community [364]. Many terms are used with different meanings; therefore, it is challenging to guarantee effective communication for describing the models and the studies and presenting the results. To overcome this limitation, in this document, we defined all the terminology before using it. However, it is necessary for a global effort of the conversational research community to define a reference vocabulary to describe research in the field.

In the future, we aim to overcome these limitations and study the long-term benefits of multimodality in the interaction. We also aim to investigate further the integration of conversational agents with modalities other than the Graphical User Interface, to strengthen the models and assess how the design principles and the users' behaviors change according to new interaction channels.

In the last month we saw an exponential growth of generative conversational interfaces, based on Large Language Models [365]. These applications are disrupting the world conversational technologies, producing results that were not thinkable even few months ago

[366]. For this reason, we want to explore these applications to understand how the contributions described in this thesis apply to this new conversational paradigm.

Bibliography

[1] Leigh Clark, Nadia Pantidi, Orla Cooney, Philip Doyle, Diego Garaialde, Justin Edwards, Brendan Spillane, Emer Gilmartin, Christine Murad, Cosmin Munteanu, et al. 2019. What makes a good conversation? challenges in designing truly conversational agents. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–12.

[2] Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2021. Bot-adversarial dialogue for safe conversational agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2950–2968.

[3] [n. d.] Au-deloitte-conversational-ai.pdf. `https://www2.deloitte.com/content/dam/Deloitte/au/Documents/strategy/au-deloitte-conversational-ai.pdf`. (Accessed on 01/02/2023). ().

[4] Karsten Wenzlaff and Sebastian Spaeth. 2022. Smarter than humans? validating how openai's chatgpt model explains crowdfunding, alternative finance and community finance. *Validating how OpenAI's ChatGPT model explains Crowdfunding, Alternative Finance and Community Finance.(December 22, 2022)*.

[5] Mubin Ul Haque, Isuru Dharmadasa, Zarrin Tasnim Sworna, Roshan Namal Rajapakse, and Hussain Ahmad. 2022. " i think this is the most disruptive technology": exploring sentiments of chatgpt early adopters using twitter data. *arXiv preprint arXiv:2212.05856*.

[6] Robbert-Jan Beun, Eveliene de Vos, and Cilia Witteman. 2003. Embodied conversational agents: effects on memory performance and anthropomorphisation. In *International workshop on intelligent virtual agents*. Springer, 315–319.

[7] Alice Kerry, Richard Ellis, and Susan Bull. 2008. Conversational agents in e–learning. In *International conference on innovative techniques and applications of artificial intelligence*. Springer, 169–182.

[8] Juan Martínez-Miranda. 2017. Embodied conversational agents for the detection and prevention of suicidal behaviour: current applications and open challenges. *Journal of medical systems*, 41, 9, 1–14.

[9] Dominic W Massaro, Ying Liu, Trevor H Chen, and Charles Perfetti. 2006. A multilingual embodied conversational agent for tutoring speech and language learning. In *INTERSPEECH*.

[10] Alastair van Heerden, Xolani Ntinga, and Khanya Vilakazi. 2017. The potential of conversational agents to provide a rapid hiv counseling and testing services. In *2017 international conference on the frontiers and advances in data science (FADS)*. IEEE, 80–85.

[11] Pietro Crovari, Sara Pidò, Pietro Pinoli, Anna Bernasconi, Arif Canakoglu, Franca Garzotto, and Stefano Ceri. 2021. Gecoagent: a conversational agent for empowering genomic data extraction and analysis. *ACM Trans. Comput. Healthcare*, 3, 1,

Article 3, (October 2021), 29 pages. ISSN: 2691-1957. DOI: 10.1145/3464383. https://doi.org/10.1145/3464383.

[12] Sara Pidò, Pietro Crovari, and Franca Garzotto. 2021. Modelling the bioinformatics tertiary analysis research process. *BMC bioinformatics*, 22, 13, 1–27.

[13] Pietro Crovari, Sara Pidò, and Franca Garzotto. 2020. Towards an ontology for tertiary bioinformatics research process. In *International Conference on Conceptual Modeling*. Springer, 82–91.

[14] Pietro Crovari, Fabio Catania, Pietro Pinoli, Philipp Roytburg, Asier Salzar, Franca Garzotto, and Stefano Ceri. 2020. Ok, dna! a conversational interface to explore genomic data. In *Proceedings of the 2nd Conference on Conversational User Interfaces*, 1–3.

[15] Sara Pidó, Pietro Pinoli, Pietro Crovari, Francesca Ieva, Franca Garzotto, and Stefano Ceri. 2023. Ask your data—supporting data science processes by combining automl and conversational interfaces. *IEEE Access*, 11, 45972–45988. DOI: 10.1109/ACCESS.2023.3272503.

[16] Giulio Antonio Abbo, Pietro Crovari, Sara Pidò, Pietro Pinoli, and Franca Garzotto. 2022. Mctk: a multi-modal conversational troubleshooting kit for supporting users in web applications. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*, 1–3.

[17] Giulio Antonio Abbo, Pietro Crovari, and Franca Garzotto. 2022. Enhancing conversational troubleshooting with multi-modality: design and implementation. In *International Workshop on Chatbot Research and Design*. Springer, to appear.

[18] Pietro Crovari, Sara Pidó, Franca Garzotto, and Stefano Ceri. 2020. Show, don't tell. reflections on the design of multi-modal conversational interfaces. In *International Workshop on Chatbot Research and Design*. Asbjørn Følstad, Theo Araujo, Symeon Papadopoulos, Effie L.-C. Law, Ewa Luger, Morten Goodwin, and Petter Bae Brandtzaeg, editors. Springer. Springer International Publishing, Cham, (November 2020), 64–77. ISBN: 978-3-030-68288-0. DOI: 10.1007/978-3-030-68288-0_5.

[19] Stephen A White. 2004. Introduction to bpmn. *Ibm Cooperation*, 2, 0, 0.

[20] Victor A McKusick and Frank H Ruddle. 1987. Toward a complete map of the human genome. *Genomics*, 1, 2, 103–106.

[21] Jun Zhang, Rod Chiodini, Ahmed Badr, and Genfa Zhang. 2011. The impact of next-generation sequencing on genomics. *Journal of genetics and genomics*, 38, 3, 95–109.

[22] Sam Behjati and Patrick S Tarpey. 2013. What is next generation sequencing? *Archives of Disease in Childhood-Education and Practice*, 98, 6, 236–238.

[23] CL Philip Chen and Chun-Yang Zhang. 2014. Data-intensive applications, challenges, techniques and technologies: a survey on big data. *Information sciences*, 275, 314–347.

[24] Rute Pereira, Jorge Oliveira, and Mário Sousa. 2020. Bioinformatics and computational tools for next-generation sequencing analysis in clinical genetics. *Journal of clinical medicine*, 9, 1, 132.

[25] Marco Masseroli, Arif Canakoglu, Pietro Pinoli, Abdulrahman Kaitoua, Andrea Gulino, Olha Horlova, Luca Nanni, Anna Bernasconi, Stefano Perna, Eirini Stamoulakatou, et al. 2019. Processing of big heterogeneous genomic datasets for tertiary analysis of next generation sequencing data. *Bioinformatics*, 35, 5, 729–736.

[26] Scott D Kahn. 2011. On the future of genomic data. *science*, 331, 6018, 728–729.

[27] A. Bernasconi, A. Canakoglu, M. Masseroli, and S. Ceri. 2020. Meta-base: a novel architecture for large-scale genomic metadata integration. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1–1.

[28] Arif Canakoglu, Anna Bernasconi, Andrea Colombo, Marco Masseroli, and Stefano Ceri. 2019. Genosurf: metadata driven semantic search system for integrated genomic datasets. *Database: The Journal of Biological Databases and Curation*, 2019.

[29] Marco Masseroli, Pietro Pinoli, Francesco Venco, Abdulrahman Kaitoua, Vahid Jalili, Fernando Palluzzi, Heiko Muller, and Stefano Ceri. 2015. GenoMetric Query Language: a novel approach to large-scale genomic data management. *Bioinformatics*, 31, 12, 1881–1888.

[30] Jeremy Leipzig. 2017. A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics*, 18, 3, 530–536.

[31] Davide Bolchini, Anthony Finkelstein, Vito Perrone, and Sylvia Nagl. 2009. Better bioinformatics through usability analysis. *Bioinformatics*, 25, 3, 406–412.

[32] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. 2016. *Jupyter Notebooks-a publishing format for reproducible computational workflows.* Volume 2016.

[33] Jeffrey S Racine. 2012. Rstudio: a platform-independent ide for r and sweave. (2012).

[34] Janez Demšar et al. 2013. Orange: data mining toolbox in python. *Journal of Machine Learning Research*, 14, 2349–2353. `http://jmlr.org/papers/v14/demsar13a.html`.

[35] Mary J Goldman, Brian Craft, Mim Hastie, Kristupas Repečka, Fran McDade, Akhil Kamath, Ayan Banerjee, Yunhai Luo, Dave Rogers, Angela N Brooks, et al. 2020. Visualizing and interpreting cancer genomics data via the xena platform. *Nature Biotechnology*, 1–4.

[36] Enis Afgan, Dannon Baker, Bérénice Batut, Marius Van Den Beek, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Björn A Grüning, et al. 2018. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic acids research*, 46, W1, W537–W544.

[37] Ravi K Madduri, Dinanath Sulakhe, Lukasz Lacinski, Bo Liu, Alex Rodriguez, Kyle Chard, Utpal J Dave, and Ian T Foster. 2014. Experiences building globus genomics: a next-generation sequencing analysis service using galaxy, globus, and amazon web services. *Concurrency and Computation: Practice and Experience*, 26, 13, 2266–2279.

[38] Michael Reich, Ted Liefeld, Joshua Gould, Jim Lerner, Pablo Tamayo, and Jill P Mesirov. 2006. Genepattern 2.0. *Nature genetics*, 38, 5, 500–501.

[39] Aaron R Quinlan and Ira M Hall. 2010. Bedtools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26, 6, 841–842.

[40] John W Nicol, Gregg A Helt, Steven G Blanchard Jr, Archana Raja, and Ann E Loraine. 2009. The integrated genome browser: free software for distribution and exploration of genome-scale datasets. *Bioinformatics*, 25, 20, 2730–2731.

[41] Robert C Gentleman, Vincent J Carey, Douglas M Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, et al. 2004. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, 5, 10, R80.

[42] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G Karahalios. 2015. Datatone: managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, 489–500.

[43] Kedar Dhamdhere, Kevin S McCurley, Ralfi Nahmias, Mukund Sundararajan, and Qiqi Yan. 2017. Analyza: exploring data with conversation. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, 493–504.

[44] Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. 2017. Applying pragmatics principles for interaction with visual analytics. *IEEE transactions on visualization and computer graphics*, 24, 1, 309–318.

[45] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M Mitchell, and Brad A Myers. 2019. Pumice: a multi-modal agent that learns concepts and conditionals from natural language and demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (UIST '19). Association for Computing Machinery, New Orleans, LA, USA, (October 2019), 577–589. ISBN: 9781450368162. DOI: 10.1145/3332165.3347899.

[46] Adam Blum. 1999. Microsoft english query 7.5: automatic extraction of semantics from relational databases and olap cubes. In *VLDB*. Volume 99, 247–248.

[47] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, 149–157.

[48] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R Mittal, and Fatma Özcan. 2016. Athena: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment*, 9, 12, 1209–1220.

[49] Antonio Messina, Agnese Augello, Giovanni Pilato, and Riccardo Rizzo. 2017. Biographbot: a conversational assistant for bioinformatics graph databases. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. Springer, 135–146.

[50] Richard S Wallace. 2009. The anatomy of alice. In *Parsing the Turing Test*. Springer, 181–210.

[51] Antonino Fiannaca, Massimo La Rosa, Laura La Paglia, Antonio Messina, and Alfonso Urso. 2016. Biographdb: a new graphdb collecting heterogeneous data for bioinformatics analysis. *Proceedings of BIOTECHNO*.

[52] Marko A Rodriguez. 2015. The gremlin graph traversal machine and language (invited talk). In *Proceedings of the 15th Symposium on Database Programming Languages*, 1–10.

[53] Walter Ritzel Paixão-Côrtes, Vanessa Stangherlin Machado Paixão-Côrtes, Cristiane Ellwanger, and Osmar Norberto de Souza. 2019. Development and usability evaluation of a prototype conversational interface for biological information retrieval via bioinformatics. In *International Conference on Human-Computer Interaction*. Springer, 575–593.

[54] P Rodriguez-Tom√©. 1998. The biocatalog. *Bioinformatics (Oxford, England)*, 14, 5, 469–470.

[55] Ethan Fast, Binbin Chen, Julia Mendelsohn, Jonathan Bassen, and Michael S Bernstein. 2018. Iris: a conversational agent for complex tasks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, (April 2018), 1–12.

[56] Rogers Jeffrey Leo John, Navneet Potti, and Jignesh M Patel. 2017. Ava: from data to insights through conversations. In *CIDR*. CIDR.

[57] Norbert E Fuchs and Rolf Schwitter. 1995. Specifying logic programs in controlled natural language. *arXiv preprint cmp-lg/9507009*.

[58] Daniel Vanderveken. 1990. *Meaning and speech acts: Volume 1, principles of language use*. Volume 1. Cambridge University Press.

[59] David Benyon and Dianne Murray. 1993. Applying user modeling to human-computer interaction design. *Artificial Intelligence Review*, 7, 3-4, 199–225.

[60] Mourad Abed, Dimitri Tabary, and Christophe Kolski. 2003. Using formal specification techniques for the modelling of tasks and generation of hci specifications. *The handbook of task analysis for human computer interaction*, 503–529.

[61] Bonnie E John and David E Kieras. 1996. The goms family of user interface analysis techniques: comparison and contrast. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 3, 4, 320–351.

[62] Egbert Schlungbaum. 1998. Support of task-based user interface design in tadeus. *Universitat Rostock*.

[63] Angel R Puerta. 1996. The mecano project: enabling user-task automation during interface development. In *Proceedings of AAAI*. Volume 96, 117–121.

[64] François Bodar, Anne-Marie Hennebert, Jean-Marie Leheureux, Isabelle Provot, Jean Vanderdonckt, and Giovanni Zucchinetti. 1996. Key activities for a development methodology of interactive applications. In *Critical Issues in User Interface Systems Engineering*. Springer, 109–134.

[65] Angel R Puerta and David Maulsby. 1997. Management of interface design knowledge with mobi-d. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, 249–252.

[66] Fabio Paternò, C Santoro, and LD Spano. 2012. Concur task trees (ctt). *Retrieved July*, 29, 2016.

[67] Li Ding, Pranam Kolari, Zhongli Ding, and Sasikanth Avancha. 2007. Using ontologies in the semantic web: a survey. In *Ontologies*. Springer, 79–113.

[68] Balakrishnan Chandrasekaran, John R Josephson, and V Richard Benjamins. 1999. What are ontologies, and why do we need them? *IEEE Intelligent Systems and their applications*, 14, 1, 20–26.

[69] Patricia G. Baker, Carole A. Goble, Sean Bechhofer, Norman W. Paton, Robert Stevens, and Andy Brass. 1999. An ontology for bioinformatics applications. *Bioinformatics (Oxford, England)*, 15, 6, 510–520.

[70] Steffen Schulze-Kremer. 2002. Ontologies for molecular biology and bioinformatics. *In Silico Biology*, 2, 3, 179–193.

[71] Anita Bandrowski, Ryan Brinkman, Mathias Brochhausen, Matthew H Brush, Bill Bug, Marcus C Chibucos, Kevin Clancy, Mélanie Courtot, Dirk Derom, Michel Dumontier, et al. 2016. The ontology for biomedical investigations. *PloS one*, 11, 4, e0154556.

[72] Robert Arp, Barry Smith, and Andrew D Spear. 2015. *Building ontologies with basic formal ontology*. Mit Press.

[73] Werner Ceusters. 2012. An information artifact ontology perspective on data collections and associated representational artifacts. In *MIE*, 68–72.

[74] Frank Kramer and Tim Beißbarth. 2017. Working with ontologies. In *Bioinformatics*. Springer, 123–135.

[75] Mario Cannataro and Pierangelo Veltri. 2007. Ms-analyzer: preprocessing and data mining services for proteomics applications on the grid. *Concurrency and Computation: practice and experience*, 19, 15, 2047–2066.

[76] Marco Masseroli, Abdulrahman Kaitoua, Pietro Pinoli, and Stefano Ceri. 2016. Modeling and interoperability of heterogeneous genomic big data for integrative processing and querying. *Methods*, 111, 3–11.

[77] Anna Bernasconi, Stefano Ceri, Alessandro Campi, and Marco Masseroli. 2017. Conceptual modeling for genomics: building an integrated repository of open data. In *Conceptual Modeling*. Heinrich C. Mayr, Giancarlo Guizzardi, Hui Ma, and Oscar Pastor, editors. Springer International Publishing, Cham, 325–339.

[78] Anna Bernasconi, Arif Canakoglu, and Stefano Ceri. 2019. From a conceptual model to a knowledge graph for genomic datasets. In *Conceptual Modeling*. Alberto H. F. Laender, Barbara Pernici, Ee-Peng Lim, and José Palazzo M. de Oliveira, editors. Springer International Publishing, Cham, 352–360.

[79] Gavin R Oliver, Steven N Hart, and Eric W Klee. 2015. Bioinformatics for clinical next generation sequencing. *Clinical chemistry*, 61, 1, 124–135.

[80] Andrew Shepherd. 1998. Hta as a framework for task analysis. *Ergonomics*, 41, 11, 1537–1552.

[81] Jeff Z Pan and OWL Working Group. 2009. Owl 2 web ontology language document overview: w3c recommendation 27 october 2009. English. (2009).

[82] John Annett. 2003. Hierarchical task analysis. In *Handbook of cognitive task design*. CRC Press, 17–36.

[83] John Annett. 2004. Hierarchical task analysis. *The handbook of task analysis for human-computer interaction*, 667.

[84] Dan Bohus and Alexander Rudnicky. 2003. Ravenclaw: dialog management using hierarchical task decomposition and an expectation agenda.

[85] Shaoke Lou, Tianxiao Li, Daniel Spakowicz, Xiting Yan, Geoffrey Lowell Chupp, and Mark Gerstein. 2020. Latent-space embedding of expression data identifies gene signatures from sputum samples of asthmatic patients. *BMC bioinformatics*, 21, 1, 1–13.

[86] Talip Zengin and Tuğba Önal-Süzek. 2020. Analysis of genomic and transcriptomic variations as prognostic signature for lung adenocarcinoma. *BMC bioinformatics*, 21, 14, 1–28.

[87] Gaia Ceddia, Liuba Nausicaa Martino, Alice Parodi, Piercesare Secchi, Stefano Campaner, and Marco Masseroli. 2020. Association rule mining to identify transcription factor interactions in genomic regions. *Bioinformatics*, 36, 4, 1007–1013.

[88] Arif Canakoglu, Luca Nanni, Artur Sokolovsky, and Stefano Ceri. 2018. Designing and evaluating deep learning models for cancer detection on gene expression data. In *International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*. Springer, 249–261.

[89] Francisco Cristovao, Silvia Cascianelli, Arif Canakoglu, Mark Carman, Luca Nanni, Pietro Pinoli, and Marco Masseroli. 2020. Investigating deep learning based breast cancer subtyping using pan-cancer and multi-omic data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

[90] Gaia Ceddia, Pietro Pinoli, Stefano Ceri, and Marco Masseroli. 2020. Matrix factorization-based technique for drug repurposing predictions. *IEEE Journal of Biomedical and Health Informatics*.

[91] Pietro Pinoli, Davide Chicco, and Marco Masseroli. 2014. Latent dirichlet allocation based on gibbs sampling for gene function prediction. In *2014 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology*. IEEE, 1–8.

[92] Luca Nanni, Stefano Ceri, and Colin Logie. 2020. Spatial patterns of ctcf sites define the anatomy of tads and their boundaries. *Genome biology*, 21, 1, 1–25.

[93] Stefano Perna, Pietro Pinoli, Stefano Ceri, and Limsoon Wong. 2020. Nautica: classifying transcription factor interactions by positional and protein-protein interaction information. *Biology direct*, 15, 1, 1–18.

[94] Gaia Ceddia, Sara Pidò, and Marco Masseroli. 2020. Network modeling and analysis of normal and cancer gene expression data. In *Computational Intelligence Methods for Bioinformatics and Biostatistics*. Paolo Cazzaniga, Daniela Besozzi, Ivan Merelli, and Luca Manzoni, editors. Springer International Publishing, Cham, 257–270. ISBN: 978-3-030-63061-4.

[95] Eleonora Cappelli, Giovanni Felici, and Emanuel Weitschek. 2018. Combining dna methylation and rna sequencing data of cancer for supervised knowledge extraction. *BioData mining*, 11, 1, 22.

[96] Fabrizio Frasca, Matteo Matteucci, Marco Masseroli, and Marco Morelli. 2018. Modeling gene transcriptional regulation by means of hyperplanes genetic clustering. In *2018 International joint conference on neural networks (IJCNN)*. IEEE, 1–8.

[97] Guray Golcuk, Mustafa Anil Tuncel, and Arif Canakoglu. 2018. Exploiting ladder networks for gene expression classification. In *International Conference on Bioinformatics and Biomedical Engineering*. Springer, 270–278.

[98] Wei Wang, Xi Yang, Chengkun Wu, and Canqun Yang. 2020. Cginet: graph convolutional network-based model for identifying chemical-gene interaction in an integrated multi-relational graph. *BMC bioinformatics*, 21, 1, 1–17.

[99] Yu Zhang, Yahui Long, and Chee Keong Kwoh. 2020. Deep learning based dna: rna triplex forming potential prediction. *BMC bioinformatics*, 21, 1, 1–13.

[100] Jun Wang and Liangjiang Wang. 2020. Prediction and prioritization of autism-associated long non-coding rnas using gene expression and sequence features. *BMC bioinformatics*, 21, 1, 1–15.

[101] Fang Jing, Shao-Wu Zhang, and Shihua Zhang. 2020. Prediction of enhancer–promoter interactions using the cross-cell type information and domain adversarial neural network. *BMC bioinformatics*, 21, 1, 1–16.

[102] Umair Ayub, Imran Haider, and Hammad Naveed. 2020. Salign–a structure aware method for global ppi network alignment. *BMC bioinformatics*, 21, 1, 1–18.

[103] Abhigyan Nath and André Leier. 2020. Improved cytokine–receptor interaction prediction by exploiting the negative sample space. *BMC bioinformatics*, 21, 1, 1–16.

[104] Zhilong Mi, Binghui Guo, Xiaobo Yang, Ziqiao Yin, and Zhiming Zheng. 2020. Lamp: disease classification derived from layered assessment on modules and pathways in the human gene network. *BMC bioinformatics*, 21, 1, 1–20.

[105] Hyein Seo and Dong-Ho Cho. 2020. Feature selection algorithm based on dual correlation filters for cancer-associated somatic variants. *BMC bioinformatics*, 21, 1, 1–19.

[106] Jian Zhang, Lixin Lv, Donglei Lu, Denan Kong, Mohammed Abdoh Ali Al-Alashaari, and Xudong Zhao. 2020. Variable selection from a feature representing protein sequences: a case of classification on bacterial type iv secreted effectors. *BMC bioinformatics*, 21, 1, 1–15.

[107] Lei Zhang, Bailong Liu, Zhengwei Li, Xiaoyan Zhu, Zhizhen Liang, and Jiyong An. 2020. Predicting mirna-disease associations by multiple meta-paths fusion graph embedding model. *BMC bioinformatics*, 21, 1, 1–19.

[108] Xingyu Zheng, Christopher I Amos, and H Robert Frost. 2020. Cancer prognosis prediction using somatic point mutation and copy number variation data: a comparison of gene-level and pathway-based models. *BMC bioinformatics*, 21, 1, 1–19.

[109] Tian-Ru Wu, Meng-Meng Yin, Cui-Na Jiao, Ying-Lian Gao, Xiang-Zhen Kong, and Jin-Xing Liu. 2020. Mccmf: collaborative matrix factorization based on matrix completion for predicting mirna-disease associations. *BMC bioinformatics*, 21, 1, 1–22.

[110] Pasan C Fernando, Paula M Mabee, and Erliang Zeng. 2020. Integration of anatomy ontology data with protein-protein interaction networks improves the candidate gene prediction accuracy for anatomical entities. *bioRxiv*.

[111] Chunxiang Wang, Xin Gao, and Juntao Liu. 2020. Impact of data preprocessing on cell-type clustering based on single-cell rna-seq data. *BMC bioinformatics*, 21, 1, 1–13.

[112] Liang-Rui Ren, Ying-Lian Gao, Jin-Xing Liu, Junliang Shang, and Chun-Hou Zheng. 2020. Correntropy induced loss based sparse robust graph regularized extreme learning machine for cancer classification. *BMC bioinformatics*, 21, 1, 1–22.

[113] Evan A Clayton, Toyya A Pujol, John F McDonald, and Peng Qiu. 2020. Leveraging tcga gene expression data to build predictive models for cancer drug response. *BMC bioinformatics*, 21, 14, 1–11.

[114] Nahim Adnan, Chengwei Lei, and Jianhua Ruan. 2020. Robust edge-based biomarker discovery improves prediction of breast cancer metastasis. *BMC bioinformatics*, 21, 14, 1–18.

[115] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. 2007. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25, 11, 1251–1255.

[116] Eric Miller. 1998. An introduction to the resource description framework. *Bulletin of the American Society for Information Science and Technology*, 25, 1, 15–19.

[117] Dan Brickley, Ramanathan V Guha, and Andrew Layman. 1999. Resource description framework (rdf) schema specification. (1999).

[118] Tania Tudorache, Jennifer Vendetti, and Natalya Fridman Noy. 2008. Web-protege: a lightweight owl ontology editor for the web. In *OWLED*. Volume 432, 2009.

[119] Virginia Braun and Victoria Clarke. 2012. *Thematic analysis.* American Psychological Association.

[120] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: open source language understanding and dialogue management. *ArXiv*, abs/1712.05181.

[121] Sumit Raj. 2018. Building chatbots with python. *Using Natural Language Processing and Machine Learning. Apress.*

[122] Ted Boren and Judith Ramey. 2000. Thinking aloud: reconciling theory and practice. *IEEE transactions on professional communication*, 43, 3, 261–278.

[123] Jeff Sauro PhD. [n. d.] Measuring Usability with the System Usability Scale (SUS) – MeasuringU. en-US. (). Retrieved 11/29/2022 from `https://measuringu.com/sus/`.

[124] Quentin Limbourg and Jean Vanderdonckt. 2004. Comparing task models for user interface design. *The handbook of task analysis for human-computer interaction*, 6, 135–154.

[125] Sharon Oviatt. 1999. Ten myths of multimodal interaction. *Communications of the ACM*, 42, 11, (November 1999), 74–81. ISSN: 0001-0782. DOI: `10.1145/319382.319398`. `https://doi.org/10.1145/319382.319398`.

[126] Abbas Mehrabi Boshrabadi and Reza Biria. 2014. The efficacy of multimodal vs. print-based texts for teaching reading comprehension skills to iranian high school third graders. *International Journal of Language Learning and Applied Linguistics World*, 5, (January 2014), 17.

[127] Rangina Ahmad, Dominik Siemon, Daniel Fernau, and Susanne Robra-Bissantz. 2020. Introducing" raffi": a personality adaptive conversational agent. In *PACIS*, 28.

[128] Rangina Ahmad, Dominik Siemon, Ulrich Gnewuch, and Susanne Robra-Bissantz. 2022. Designing personality-adaptive conversational agents for mental health care. *Information Systems Frontiers*, 24, 3, 923–943.

[129] Ricarda Schlimbach, Heidi Rinn, Daniel Markgraf, and Susanne Robra-Bissantz. 2022. A literature review on pedagogical conversational agent adaptation. In *Pacific Asia Conference on Information Systems*, 1.

[130] Krzysztof Witkowski. 2017. Internet of things, big data, industry 4.0–innovative solutions in logistics and supply chains management. *Procedia engineering*, 182, 763–769.

[131] Fionn Murtagh and Keith Devlin. 2018. The development of data science: implications for education, employment, research, and the data revolution for sustainable development. *Big Data and Cognitive Computing*, 2, 2, 14.

[132] James Honaker and Vito D ' Orazio. [n. d.] Statistical modeling by gesture: a graphical, browser-based statistical interface for data repositories. `http://ceur-ws.org/Vol-1210/datawiz2014_05.pdf`. Accessed: 2021-12-31. ().

[133] C V Krishnakumar Iyer, Feili Hou, Henry Wang, Yonghong Wang, Kay Oh, Swetava Ganguli, and Vipul Pandey. 2021. Trinity: a No-Code AI platform for complex spatial datasets. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery* (GEOAI '21). Association for Computing Machinery, Beijing, China, (November 2021), 33–42. ISBN: 9781450391207. DOI: `10.1145/3486635.3491072`.

[134] David Price, Ellen Riloffff, Joseph Zachary, and Brandon Harvey. 2000. Naturaljava: a natural language interface for programming in java. In *Proceedings of the 5th international conference on Intelligent user interfaces*, 207–211.

[135] Miltiadis Allamanis, Earl T Barr, Premkumar Devanbu, and Charles Sutton. 2018. A survey of machine learning for big code and naturalness. *ACM Computing Surveys (CSUR)*, 51, 4, 1–37.

[136] VG Renumol, Dharanipragada Janakiram, and S Jayaprakash. 2010. Identification of cognitive processes of effective and ineffective students during computer programming. *ACM Transactions on Computing Education (TOCE)*, 10, 3, 1–21.

[137] Jiho ShinOzan2021 and Jaechang Nam. 2021. A survey of automatic code generation from natural language.*Journal of Information Processing Systems*, 17, 3, (June 2021). ISSN: 1976-913X. DOI: `10.3745/JIPS.04.0216`.

[138] Stephen Chong and Riccardo Pucella. 2004. A framework for creating natural language user interfaces for action-based applications. *arXiv preprint cs/0412065*.

[139] David Vadas and James R Curran. 2005. Programming with unrestricted natural language. In *Proceedings of the Australasian Language Technology Workshop 2005*, 191–199.

[140] Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696*.

[141] Henry Lieberman and Moin Ahmad. 2010. Knowing what you're talking about: natural language programming of a multi-player online game. In *No Code Required*. Elsevier, 331–343.

[142] Vu Le, Sumit Gulwani, and Zhendong Su. 2013. Smartsynth: synthesizing smartphone automation scripts from natural language. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, 193–206.

[143] Xi Victoria Lin, Chenglong Wang, Deric Pang, Kevin Vu, and Michael D Ernst. 2017. Program synthesis from natural language using recurrent neural networks. *University of Washington Department of Computer Science and Engineering, Seattle, WA, USA, Tech. Rep. UW-CSE-17-03-01*.

[144] Jinfeng Lin, Yalin Liu, Qingkai Zeng, Meng Jiang, and Jane Cleland-Huang. 2021. Traceability transformed: generating more accurate links with pre-trained bert models. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 324–335.

[145] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

[146] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. Automl: a survey of the state-of-the-art. *Knowledge-Based Systems*, 212, 106622.

[147] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2019. Auto-sklearn: efficient and robust automated machine learning. In *Automated Machine Learning*. Springer, Cham, 113–134.

[148] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 847–855.

[149] Randal S Olson and Jason H Moore. 2016. Tpot: a tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*. PMLR, 66–74.

[150] Shubhra Kanti Karmaker, Md Mahadi Hassan, Micah J Smith, Lei Xu, Chengxiang Zhai, and Kalyan Veeramachaneni. 2021. Automl to date and beyond: challenges and opportunities. *ACM Computing Surveys (CSUR)*, 54, 8, 1–36.

[151] Djallel Bouneffouf, Charu Aggarwal, Thanh Hoang, Udayan Khurana, Horst Samulowitz, Beat Buesser, Sijia Liu, Tejaswini Pedapati, Parikshit Ram, Ambrish Rawat, et al. 2020. Survey on automated end-to-end data science? In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–9.

[152] Tijl De Bie, Luc De Raedt, José Hernández-Orallo, Holger H Hoos, Padhraic Smyth, and Christopher KI Williams. 2021. Automating data science: prospects and challenges. *arXiv preprint arXiv:2105.05699*.

[153] Erol Ozan. 2021. A novel browser-based no-code machine learning application development tool. In *2021 IEEE World AI IoT Congress (AIIoT)*. (May 2021), 0282–0284. DOI: 10.1109/AIIoT52608.2021.9454239.

[154] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, et al. 2019. Tensorflow. js: machine learning for the web and beyond. *arXiv preprint arXiv:1901.05350*.

[155] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. 2017. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*. Ieee, 1–6.

[156] Michelle Carney, Barron Webster, Irene Alvarado, Kyle Phillips, Noura Howell, Jordan Griffith, Jonas Jongejan, Amit Pitaru, and Alexander Chen. 2020. Teachable machine: approachable web-based tool for exploring machine learning classification. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*, 1–8.

[157] Shweta Narkar, Yunfeng Zhang, Q Vera Liao, Dakuo Wang, and Justin D Weisz. 2021. Model lineupper: supporting interactive model comparison at multiple levels for automl. In *26th International Conference on Intelligent User Interfaces*, 170–174.

[158] Remco Chang. 2021. Snowcat and CAVA: Visualization Tools for Interacting with AutoML and Knowledgebases. Technical report. TUFTS UNIV.

[159] Dakuo Wang, Josh Andres, Justin D Weisz, Erick Oduor, and Casey Dugan. 2021. AutoDS: towards Human-Centered automation of data science. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, (May 2021), 1–12. ISBN: 9781450380966. DOI: 10.1145/3411764.3445526.

[160] Linhao Meng, Stef Van Den Elzen, and Anna Vilanova. 2022. Modelwise: interactive model comparison for model diagnosis, improvement and selection. In *Computer Graphics Forum* number 3. Volume 41. Wiley Online Library, 97–108.

[161] Gary King. 2007. An introduction to the dataverse network as an infrastructure for data sharing. (2007).

[162] Arun S Maiya. 2020. Ktrain: a Low-Code library for augmented machine learning. *arXiv preprint arXiv:2004.10703*.

[163] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. 2018. Metrics for explainable ai: challenges and prospects. *arXiv preprint arXiv:1812.04608*.

[164] David Heckerman, John S. Breese, and Koos Rommelse. 1995. Decision-theoretic troubleshooting. *Communications of the ACM*, 38, 3, (March 1995), 49–57. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/203330.203341.

[165] Camilo Thorne. 2017. Chatbots for troubleshooting: A survey. *Language and Linguistics Compass*, 11, 10, e12253. ISSN: 1749-818X. DOI: 10.1111/lnc3.12253.

[166] Kalyan Moy Gupta. 2001. Taxonomic Conversational Case-Based Reasoning. In *Case-Based Reasoning Research and Development* (Lecture Notes in Computer Science). David W. Aha and Ian Watson, editors. Springer, Berlin, Heidelberg, 219–233. ISBN: 978-3-540-44593-7. DOI: 10.1007/3-540-44593-5_16.

[167] Guoguang Zhao, Jianyu Zhao, Yang Li, Christoph Alt, Robert Schwarzenberg, Leonhard Hennig, Stefan Schaffer, Sven Schmeier, Changjian Hu, and Feiyu Xu.

2019. MOLI: Smart Conversation Agent for Mobile Customer Service. *Information*, 10, 2, (February 2019), 63. DOI: 10.3390/info10020063.

[168] Sethuramalingam Subramaniam, Pooja Aggarwal, Gargi B. Dasgupta, and Amit Paradkar. 2018. COBOTS - A Cognitive Multi-Bot Conversational Framework for Technical Support. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (AAMAS '18). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, (July 2018), 597–604.

[169] Oriol Vinyals and Quoc Le. 2015. A Neural Conversational Model. (July 2015). DOI: 10.48550/arXiv.1506.05869. arXiv: 1506.05869 [cs].

[170] Asbjørn Følstad, Cecilie Bertinussen Nordheim, and Cato Alexander Bjørkli. 2018. What Makes Users Trust a Chatbot for Customer Service? An Exploratory Interview Study. In *Internet Science* (Lecture Notes in Computer Science). Svetlana S. Bodrunova, editor. Springer International Publishing, Cham, 194–208. ISBN: 978-3-030-01437-7. DOI: 10.1007/978-3-030-01437-7_16.

[171] Sharon Oviatt. 2007. Multimodal Interfaces. In *The Human-Computer Interaction Handbook*. (Second edition). CRC Press. ISBN: 978-0-429-16397-5.

[172] Dominic W. Massaro. 2004. A framework for evaluating multimodal integration by humans and a role for embodied conversational agents. In *Proceedings of the 6th International Conference on Multimodal Interfaces - ICMI '04*. ACM Press, State College, PA, USA, 24. ISBN: 978-1-58113-995-2. DOI: 10.1145/1027933.1027939.

[173] Sharon Oviatt, Rachel Coulston, and Rebecca Lunsford. 2004. When do we interact multimodally? Cognitive load and multimodal communication patterns. *Proceedings of the 6th international conference on Multimodal interfaces*, 8.

[174] Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems*, 31, 6, 5–14.

[175] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

[176] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[177] Guillaume Klein, Yoon Kim, Yuntian Deng, Vincent Nguyen, Jean Senellart, and Alexander M Rush. 2018. Opennmt: neural machine translation toolkit. *arXiv preprint arXiv:1805.11462*.

[178] Xiaofeng Dai, Ting Li, Zhonghu Bai, Yankun Yang, Xiuxia Liu, Jinling Zhan, and Bozhi Shi. 2015. Breast cancer intrinsic subtype classification, clinical use and future trends. *American journal of cancer research*, 5, 10, 2929.

[179] kaggle. [n. d.] Stroke prediction dataset. https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset. [Online; accessed 28-April-2022]. ().

[180] john Brooke. 1996. SUS: a 'Quick and Dirty' Usability Scale. In *Usability Evaluation In Industry*. CRC Press. ISBN: 978-0-429-15701-1.

[181] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): results of Empirical and Theoretical Research. In *Advances in Psychology*. Human Mental Workload. Volume 52. North-Holland, (January 1, 1988), 139–183. DOI: `10.1016/S0166-4115(08)62386-9`.

[182] Sandra G. Hart. 2006. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50, 9, (October 1, 2006), 904–908. DOI: `10/fzvtd4`.

[183] Alberto Benayas, Miguel Angel Sicilia, and Marçal Mora-Cantallops. 2023. Automated creation of an intent model for conversational agents. *Applied Artificial Intelligence*, 37, 1, 2164401.

[184] Ali Ahmadvand. 2020. User intent inference for web search and conversational agents. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 911–912.

[185] John Brooke. 2013. Sus: a retrospective. *Journal of usability studies*, 8, 2, 29–40.

[186] Rebecca A. Grier. 2015. How High is High? a Meta-Analysis of NASA-TLX Global Workload Scores. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 59, 1, (September 2015), 1727–1731. DOI: `10/gf447w`.

[187] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6, 52138–52160.

[188] Q Vera Liao, Daniel Gruen, and Sarah Miller. 2020. Questioning the ai: informing design practices for explainable ai user experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–15.

[189] Zheng Zhang, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu. 2020. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, 63, 10, 2011–2027.

[190] Yinong Long, Jianan Wang, Zhen Xu, Zongsheng Wang, Baoxun Wang, and Zhuoran Wang. 2017. A knowledge enhanced generative conversational service agent. In *Proceedings of the 6th Dialog System Technology Challenges (DSTC6) Workshop*.

[191] Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. 2020. Challenges in building intelligent open-domain dialog systems. *ACM Transactions on Information Systems (TOIS)*, 38, 3, 1–32.

[192] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19, 2, 25–35.

[193] Salla Syvänen and Chiara Valentini. 2020. Conversational agents in online organization–stakeholder interactions: a state-of-the-art analysis and implications for further research. *Journal of Communication Management*, 24, 4, 339–362.

[194] Michael McTear. 2020. Conversational ai: dialogue systems, conversational agents, and chatbots. *Synthesis Lectures on Human Language Technologies*, 13, 3, 1–251.

[195] Luis Fernando Lins, Glaucia Melo, Toacy Oliveira, Paulo Alencar, and Donald Cowan. 2021. Pacas: process-aware conversational agents. In *International Conference on Business Process Management*. Springer, 312–318.

[196] Shafquat Hussain, Omid Ameri Sianaki, and Nedal Ababneh. 2019. A survey on conversational agents/chatbots classification and design techniques. In *Workshops*

of the International Conference on Advanced Information Networking and Applications. Springer, 946–956.

[197] Marie-Luce Bourguet. 2003. Designing and prototyping multimodal commands. In *Interact*. Volume 3. Citeseer, 717–720.

[198] Sigrid Norris. 2004. *Analyzing multimodal interaction: A methodological framework*. Routledge.

[199] Matthew Turk. 2014. Multimodal interaction: a review. *Pattern recognition letters*, 36, 189–195.

[200] David Griol, Ismael Baena, José Manuel Molina, and Araceli Sanchis de Miguel. 2014. A multimodal conversational agent for personalized language learning. In *Ambient intelligence-software and applications*. Springer, 13–21.

[201] SS Muhammad Nizam, Rimaniza Zainal Abidin, Nurhazarifah Che Hashim, Meng Chun Lam, Haslina Arshad, and NAA Majid. 2018. A review of multimodal interaction technique in augmented reality environment. *Int. J. Adv. Sci. Eng. Inf. Technol*, 8, 4-2, 1460.

[202] Harry Bunt, Robbert-Jan Beun, and Tijn Borghuis. 1998. *Multimodal human-computer communication: systems, techniques, and experiments*. Volume 1374. Springer Science & Business Media.

[203] Maria Chiara Caschera, Fernando Ferri, and Patrizia Grifoni. 2007. Multimodal interaction systems: information and time features. *International Journal of Web and Grid Services*, 3, 1, 82–99.

[204] Aditya Sankar and Steven M Seitz. 2016. In situ cad capture. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, 233–243.

[205] Mirko Gelsomini, Giulia Cosentino, Micol Spitale, Mattia Gianotti, Davide Fisicaro, Giulia Leonardi, Fabiano Riccardi, Agnese Piselli, Eleonora Beccaluva, Barbara Bonadies, et al. 2019. Magika, a multisensory environment for play, education and inclusion. In *Extended abstracts of the 2019 CHI conference on human factors in computing systems*, 1–6.

[206] Pietro Crovari, Mattia Gianotti, Fabiano Riccardi, and Franca Garzotto. 2019. Designing a smart toy: guidelines from the experience with smart dolphin" sam". In *Proceedings of the 13th Biannual Conference of the Italian SIGCHI Chapter: Designing the next interaction*, 1–10.

[207] Eleni Papadaki, Stavroula Ntoa, Ilia Adami, and Constantine Stephanidis. 2017. Let's cook: an augmented reality system towards developing cooking skills for children with cognitive impairments. In *International conference on smart objects and technologies for social good*. Springer, 237–247.

[208] Lizi Liao, Yunshan Ma, Xiangnan He, Richang Hong, and Tat-seng Chua. 2018. Knowledge-aware multimodal dialogue systems. In *Proceedings of the 26th ACM international conference on Multimedia*, 801–809.

[209] Sharon Oviatt. 2003. Advances in robust multimodal interface design. *IEEE computer graphics and applications*, 5, 62–68.

[210] Deborah A Dahl. 2017. Multimodal interaction with w3c standards. *Cham, Switzerland: Springer International Publishing*.

[211] Fabio Catania. 2020. Conversational technology and natural language visualization for children's learning. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–7.

[212] Pattie Maes. 1994. Agents that reduce work and information overload. *Commun. ACM*, 37, 7, (July 1994), 30–40.

[213] E Horvitz. 1999. Principles of mixed-initiative user interfaces. *Proceedings of the SIGCHI conference on Human*.

[214] Fabio Catania, Nicola Di Nardo, Franca Garzotto, and Daniele Occhiuto. 2019. Emoty: an emotionally sensitive conversational agent for people with neurodevelopmental disorders. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*.

[215] Fabio Catania, Giorgio De Luca, Nicola Bombaci, Erica Colombo, Pietro Crovari, Eleonora Beccaluva, and Franca Garzotto. 2020. Musical and conversational artificial intelligence. In *Proceedings of the 25th International Conference on Intelligent User Interfaces Companion*, 51–52.

[216] Fabio Catania, Micol Spitale, Davide Fisicaro, and Franca Garzotto. 2019. Cork: a conversational agent framework exploiting both rational and emotional intelligence. In *IUI Workshops*.

[217] Pietro Crovari, Fabio Catania, and Franca Garzotto. 2020. Crime story as a tool for scientific and technological outreach. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–10.

[218] Claudia Maria Cutrupi, Salvatore Fadda, Giovanni Valcarenghi, Giulia Cosentino, Fabio Catania, Micol Spitale, and Franca Garzotto. 2020. Smemo: a multi-modal interface promoting children's creation of personal conversational agents. In *Proceedings of the 2nd Conference on Conversational User Interfaces*, 1–3.

[219] Amirreza Rouhi, Micol Spitale, Fabio Catania, Giulia Cosentino, Mirko Gelsomini, and Franca Garzotto. 2019. Emotify: emotional game for children with autism spectrum disorder based-on machine learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion*, 31–32.

[220] Micol Spitale, Fabio Catania, Pietro Crovari, and Franca Garzotto. 2020. Multicriteria decision analysis and conversational agents for children with autism. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*.

[221] Micol Spitale, Silvia Silleresi, Giulia Cosentino, Francesca Panzeri, and Franca Garzotto. 2020. " whom would you like to talk with?" exploring conversational agents for children's linguistic assessment. In *Proceedings of the Interaction Design and Children Conference*, 262–272.

[222] Jakob Nielsen. 2005. Ten usability heuristics. (2005).

[223] MJ Page, JE McKenzie, PM Bossuyt, I Boutron, TC Hoffmann, CD Mulrow, L Shamseer, JM Tetzlaff, EA Akl, SE Brennan, et al. [n. d.] The prisma 2020 statement: an updated guideline for reporting systematic reviews. syst rev. 2021; 10: 89. ().

[224] Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. Paradise: a framework for evaluating spoken dialogue agents. (1997). arXiv: `cmp-lg/9704004 [cmp-lg]`.

[225] Anton Chekhov. 1999. *The Unknown Chekhov: Stories & Other Writings Hitherto Untranslated*. Macmillan.

[226] Qiyu Zhi and Ronald Metoyer. 2020. Gamebot: a visualization-augmented chatbot for sports game. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–7.

[227] Marti Hearst and Melanie Tory. 2019. Would you like a chart with that? incorporating visualizations into conversational interfaces. In *2019 IEEE Visualization Conference (VIS)*. IEEE, 1–5.

[228] Larry L Constantine and Lucy AD Lockwood. 1999. *Software for use: a practical guide to the models and methods of usage-centered design*. Pearson Education.

[229] David Griol and Zoraida Callejas. 2013. An architecture to develop multimodal educative applications with chatbots. *International Journal of Advanced Robotic Systems*, 10, 3, 175.

[230] Nadine B Sarter. 2006. Multimodal information presentation: design guidance and research challenges. *International journal of industrial ergonomics*, 36, 5, 439–445.

[231] Justine Cauell, Tim Bickmore, Lee Campbell, and Hannes Vilhjálmsson. 2000. Designing embodied conversational agents. *Embodied conversational agents*, 29.

[232] Sharon Oviatt. 1997. Mulitmodal interactive maps: designing for human performance. *Human–Computer Interaction*, 12, 1-2, 93–129.

[233] Sharon Oviatt, Antonella DeAngeli, and Karen Kuhn. 1997. Integration and synchronization of input modes during multimodal human-computer interaction. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, 415–422.

[234] Stergios Tegos, Stavros Demetriadis, Georgios Psathas, and Thrasyvoulos Tsiatsos. 2019. A configurable agent to advance peers' productive dialogue in moocs. In *International Workshop on Chatbot Research and Design*. Springer, 245–259.

[235] Alice Kerlyl, Phil Hall, and Susan Bull. 2006. Bringing chatbots into education: towards natural language negotiation of open learner models. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 179–192.

[236] Bruno Dumas, Denis Lalanne, and Sharon Oviatt. 2009. Multimodal interfaces: a survey of principles, models and frameworks. In *Human machine interaction*. Springer, 3–26.

[237] Jonathan Grudin. 1989. The case against user interface consistency. *Communications of the ACM*, 32, 10, 1164–1173.

[238] Joakim Gustafson, Linda Bell, Jonas Beskow, Johan Boye, Rolf Carlson, Jens Edlund, Björn Granström, David House, and Mats Wirén. 2000. Adapt—a multimodal conversational dialogue system in an apartment domain. In *The Sixth International Conference on Spoken Language Processing (ICSLP), Beijing, China*, 134–137.

[239] Adwait Ratnaparkhi. 1998. Maximum entropy models for natural language ambiguity resolution.

[240] Raina Langevin, Ross J Lordon, Thi Avrahami, Benjamin R Cowan, Tad Hirsch, and Gary Hsieh. 2021. Heuristic evaluation of conversational agents. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–15.

[241] Andrew J Cowell and Kay M Stanney. 2003. Embodiment and interaction guidelines for designing credible, trustworthy embodied conversational agents. In *Intelligent Virtual Agents: 4th International Workshop, IVA 2003, Kloster Irsee, Germany, September 15-17, 2003. Proceedings 4*. Springer, 301–309.

[242] David R Large, Gary Burnett, and Leigh Clark. 2019. Lessons from oz: design guidelines for automotive conversational user interfaces. In *Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications: Adjunct Proceedings*, 335–340.

[243] Shivang Verma, Lakshay Sahni, and Moolchand Sharma. 2020. Comparative analysis of chatbots. In *Proceedings of the International Conference on Innovative Computing & Communications (ICICC)*.

[244] Irene Lopatovska, Katrina Rink, Ian Knight, Kieran Raines, Kevin Cosenza, Harriet Williams, Perachya Sorsche, David Hirsch, Qi Li, and Adrianna Martinez. 2019. Talk to me: exploring user interactions with the amazon alexa. *Journal of Librarianship and Information Science*, 51, 4, 984–997.

[245] Satwinder Singh and Himanshu Beniwal. 2021. A survey on near-human conversational agents. *Journal of King Saud University-Computer and Information Sciences*.

[246] Matthew L Meuter, Amy L Ostrom, Robert I Roundtree, and Mary Jo Bitner. 2000. Self-service technologies: understanding customer satisfaction with technology-based service encounters. *Journal of marketing*, 64, 3, 50–64.

[247] Mary Jo Bitner, Stephen W Brown, and Matthew L Meuter. 2000. Technology infusion in service encounters. *Journal of the Academy of marketing Science*, 28, 1, 138–149.

[248] Anne Scherer, Nancy V Wünderlich, and Florian Von Wangenheim. 2015. The value of self-service. *MIS quarterly*, 39, 1, 177–200.

[249] Mehdi Alaimi, Edith Law, Kevin Daniel Pantasdo, Pierre-Yves Oudeyer, and Hélène Sauzeon. 2020. Pedagogical agents for fostering question-asking skills in children. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–13.

[250] Jessy Ceha, Ken Jen Lee, Elizabeth Nilsen, Joslin Goh, and Edith Law. 2021. Can a humorous conversational agent enhance learning experience and outcomes? In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–14.

[251] Aditya Nrusimha Vaidyam, Hannah Wisniewski, John David Halamka, Matcheri S Kashavan, and John Blake Torous. 2019. Chatbots and conversational agents in mental health: a review of the psychiatric landscape. *The Canadian Journal of Psychiatry*, 64, 7, 456–464.

[252] Robert R Morris, Kareem Kouddous, Rohan Kshirsagar, and Stephen M Schueller. 2018. Towards an artificially empathic conversational agent for mental health applications: system design and user perceptions. *Journal of medical Internet research*, 20, 6, e10148.

[253] Liliana Laranjo, Adam G Dunn, Huong Ly Tong, Ahmet Baki Kocaballi, Jessica Chen, Rabia Bashir, Didi Surian, Blanca Gallego, Farah Magrabi, Annie YS Lau, et al. 2018. Conversational agents in healthcare: a systematic review. *Journal of the American Medical Informatics Association*, 25, 9, 1248–1258.

[254] Jean-Emmanuel Bibault, Benjamin Chaix, Pierre Nectoux, Arthur Pienkowski, Arthur Guillemasé, and Benoît Brouard. 2019. Healthcare ex machina: are conversational agents ready for prime time in oncology? *Clinical and translational radiation oncology*, 16, 55–59.

[255] Ulrich Gnewuch, Stefan Morana, and Alexander Maedche. 2017. Towards designing cooperative and social conversational agents for customer service. In *ICIS*.

[256] Alisa Kongthon, Chatchawal Sangkeettrakarn, Sarawoot Kongyoung, and Choochart Haruechaiyasak. 2009. Implementing an online help desk system based on conversational agent. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, 450–451.

[257] David Griol, José Manuel Molina, and Araceli Sanchis De Miguel. 2014. Developing multimodal conversational agents for an enhanced e-learning experience. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 3, 1, 13–26.

[258] Pietro Crovari, Sara Pidò, Pietro Pinoli, Anna Bernasconi, Arif Canakoglu, Franca Garzotto, and Stefano Ceri. 2021. GeCoAgent: a conversational agent for empowering genomic data extraction and analysis. *ACM Trans. Comput. Healthcare*, 3, 1, (October 2021), 1–29. ISSN: 2691-1957. DOI: 10.1145/3464383.

[259] Francisco AM Valério, Tatiane G Guimarães, Raquel O Prates, and Heloisa Candello. 2020. Comparing users' perception of different chatbot interaction paradigms: a case study. In *Proceedings of the 19th Brazilian Symposium on Human Factors in Computing Systems*, 1–10.

[260] Wahlster. 2003. Smartkom: symmetric multimodality in an adaptive and reusable dialogue shell. *Proceedings of the human computer interaction status*.

[261] Jasper Feine, Ulrich Gnewuch, Stefan Morana, and Alexander Maedche. 2019. A taxonomy of social cues for conversational agents. *International Journal of Human-Computer Studies*, 132, 138–161.

[262] Stephan Diederich, Alfred Benedikt Brendel, and Lutz M Kolbe. 2019. Towards a taxonomy of platforms for conversational agent design.

[263] Xi Yang and Marco Aurisicchio. 2021. Designing conversational agents: a self-determination theory approach. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–16.

[264] Philipp Kirschthaler, Martin Porcheron, and Joel E Fischer. 2020. What can i say? effects of discoverability in vuis on task performance and user experience. In *Proceedings of the 2nd Conference on Conversational User Interfaces*, 1–9.

[265] Mauajama Firdaus, Arunav Pratap Shandeelya, and Asif Ekbal. 2020. More to diverse: generating diversified responses in a task oriented multimodal dialog system. *PloS one*, 15, 11, e0241271.

[266] Toby Jia-Jun Li, Jingya Chen, Haijun Xia, Tom M Mitchell, and Brad A Myers. 2020. Multi-modal repairs of conversational breakdowns in task-oriented dialogs. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, 1094–1107.

[267] Yi Shan, Meng Ji, Wenxiu Xie, Xiaobo Qian, Rongying Li, Xiaomin Zhang, Tianyong Hao, et al. 2022. Language use in conversational agent–based health communication: systematic review. en. *Journal of Medical Internet Research*, 24, 7, (July 2022), e37403. ISSN: 1439-4456, 1438-8871. DOI: 10.2196/37403.

[268] Ryan M Schuetzler, G Mark Grimes, and Justin Scott Giboney. 2020. The impact of chatbot conversational skill on engagement and perceived humanness. *Journal of Management Information Systems*, 37, 3, 875–900.

[269] Eun Go and S Shyam Sundar. 2019. Humanizing chatbots: the effects of visual, identity and conversational cues on humanness perceptions. *Computers in Human Behavior*, 97, 304–316.

[270] Sangwon Lee, Naeun Lee, and Young June Sah. 2020. Perceiving a mind in a chatbot: effect of mind perception and social cues on co-presence, closeness, and intention to use. *International Journal of Human–Computer Interaction*, 36, 10, 930–940.

[271] Tibert Verhagen, Jaap Van Nes, Frans Feldberg, and Willemijn Van Dolen. 2014. Virtual customer service agents: using social presence and personalization to shape online service encounters. *Journal of Computer-Mediated Communication*, 19, 3, 529–545.

[272] Timothy W Bickmore and Rosalind W Picard. 2005. Establishing and maintaining long-term human-computer relationships. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12, 2, 293–327.

[273] Justine Cassell. 2000. Embodied conversational interface agents. *Communications of the ACM*, 43, 4, 70–78.

[274] Matthew D Pickard, Judee K Burgoon, and Douglas C Derrick. 2014. Toward an objective linguistic-based measure of perceived embodied conversational agent power and likeability. *International Journal of Human-Computer Interaction*, 30, 6, 495–516.

[275] Nicole Novielli, Fiorella de Rosis, and Irene Mazzotta. 2010. User attitude towards an embodied conversational agent: effects of the interaction mode. *Journal of pragmatics*, 42, 9, (September 2010), 2385–2397. ISSN: 0378-2166. DOI: 10.1016/j.pragma.2009.12.016.

[276] Leigh Clark, Philip Doyle, Diego Garaialde, Emer Gilmartin, Stephan Schlögl, Jens Edlund, Matthew Aylett, João Cabral, Cosmin Munteanu, Justin Edwards, et al. 2019. The state of speech in hci: trends, themes and challenges. *Interacting with computers*, 31, 4, 349–371.

[277] Razan Jaber and Donald McMillan. 2020. Conversational user interfaces on mobile devices: survey. In *Proceedings of the 2nd Conference on Conversational User Interfaces*, 1–11.

[278] S. Oviatt and R. VanGent. 1996. Error resolution during multimodal human-computer interaction. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96*. Volume 1. (October 1996), 204–207 vol.1. DOI: `10.1109/ICSLP.1996.607077`.

[279] Isabel Kathleen Fornell Haugeland. 2020. *Understanding the user experience of customer service chatbots An experimental study of how user experience is affected by differences in interaction design*. Master's thesis.

[280] Gregory A Sanders, Jean C Scholtz, et al. 2001. Measurement and evaluation of embodied conversational agents.

[281] Amon Rapp, Lorenzo Curti, and Arianna Boldi. 2021. The human side of human-chatbot interaction: a systematic literature review of ten years of research on text-based chatbots. *International Journal of Human-Computer Studies*, 151, 102630.

[282] Heloisa Candello, Claudio Pinhanez, Mauro Pichiliani, Paulo Cavalin, Flavio Figueiredo, Marisa Vasconcelos, and Haylla Do Carmo. 2019. The effect of audiences on the user experience with conversational interfaces in physical spaces. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–13.

[283] Sara Pérez-Soler, Esther Guerra, and Juan de Lara. 2020. Model-Driven chatbot development. In *Conceptual Modeling*. Springer International Publishing, 207–222. DOI: `10.1007/978-3-030-62522-1\_15`.

[284] Kiran Ramesh, Surya Ravishankaran, Abhishek Joshi, and K Chandrasekaran. 2017. A survey of design techniques for conversational agents. In *Information, Communication and Computing Technology*. Springer Singapore, 336–350. DOI: `10.1007/978-981-10-6544-6\_31`.

[285] Jan-Gerrit Harms, Pavel Kucherbaev, Alessandro Bozzon, and Geert-Jan Houben. 2018. Approaches for dialog management in conversational agents. *IEEE Internet Computing*, 23, 2, 13–22.

[286] Terry Winograd, Fernando Flores, and Fernando F Flores. 1986. *Understanding computers and cognition: A new foundation for design*. Intellect Books.

[287] Pauchet, El Fallah Seghrouchni, and Chaignaud. 2007. Simulating a human cooperative problem solving. *Produits pharmaceutiques*. ISSN: 0370-1565.

[288] Richard Wallace. 2003. The elements of aiml style. *Alice AI Foundation*, 139.

[289] Michele L McNeal and David Newyear. 2013. Chatbot creation options. *Library Technology Reports*, 49, 8, 11–17.

[290] Berardina De Carolis, Catherine Pelachaud, Isabella Poggi, and Mark Steedman. 2004. APML, a markup language for believable behavior generation. In *Life-Like Characters: Tools, Affective Functions, and Applications*. Helmut Prendinger and Mitsuru Ishizuka, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 65–85. ISBN: 9783662083734. DOI: `10.1007/978-3-662-08373-4\_4`.

[291] Stefan Kopp, Brigitte Krenn, Stacy Marsella, Andrew N Marshall, Catherine Pelachaud, Hannes Pirker, Kristinn R Thórisson, and Hannes Vilhjálmsson. 2006. Towards a common framework for multimodal generation: the behavior markup

language. In *Intelligent Virtual Agents*. Springer Berlin Heidelberg, 205–217. DOI: 10.1007/11821830_17.

[292] Stefan Kopp, Bernhard Jung, Nadine Lessmann, and Ipke Wachsmuth. 2003. Max-a multimodal assistant in virtual reality construction. *KI*, 17, 4, 11.

[293] Justine Cassell, Hannes Högni Vilhjálmsson, and Timothy Bickmore. 2004. Beat: the behavior expression animation toolkit. In *Life-Like Characters*. Springer, 163–185.

[294] Paul Piwek, Brigitte Krenn, Marc Schröder, Martine Grice, Stefan Baumann, and Hannes Pirker. 2004. Rrl: a rich representation language for the description of agent behaviour in neca. *arXiv preprint cs/0410022*.

[295] Zacharie Alès, Guillaume Dubuisson Duplessis, Ovidiu Şerban, and Alexandre Pauchet. 2012. A methodology to design human-like embodied conversational agents. In *International Workshop on Human-Agent Interaction Design and Models (HAIDM'12)*, online–proceedings.

[296] Shubham Agarwal, Ondrej Dusek, Ioannis Konstas, and Verena Rieser. 2018. A Knowledge-Grounded multimodal Search-Based conversational agent, (October 2018). arXiv: 1810.11954 [cs.CL].

[297] Robert J Moore. 2018. A natural conversation framework for conversational UX design. In *Studies in Conversational UX Design*. Robert J Moore, Margaret H Szymanski, Raphael Arar, and Guang-Jie Ren, editors. Springer International Publishing, Cham, 181–204. ISBN: 9783319955797. DOI: 10.1007/978-3-319-95579-7_9.

[298] Robert J Moore and Raphael Arar. 2019. *Conversational UX Design: A Practitioner's Guide to the Natural Conversation Framework*. Association for Computing Machinery, New York, NY, USA. ISBN: 9781450363013.

[299] Peter Wallis, Helen Mitchard, Jyotsna Das, and Damian O'Dea. 2001. Dialogue modelling for a conversational agent. In *Australian Joint Conference on Artificial Intelligence*. Springer, 532–544.

[300] Elena Planas, Gwendal Daniel, Marco Brambilla, and Jordi Cabot. 2021. Towards a model-driven approach for multiexperience AI-based user interfaces. *Software and Systems Modeling*, 20, 4, (August 2021), 997–1009. ISSN: 1619-1374. DOI: 10.1007/s10270-021-00904-y.

[301] Carlo Bernaschina, Sara Comai, and Piero Fraternali. 2018. Formal semantics of omg's interaction flow modeling language (ifml) for mobile and rich-client application model driven development. *Journal of Systems and Software*, 137, 239–260.

[302] Sharon Oviatt and Philip R Cohen. 2015. The paradigm shift to multimodality in contemporary computer interfaces. *Synthesis lectures on human-centered informatics*, 8, 3, 1–243.

[303] Rogers Jeffrey Leo John, Jignesh M Patel, Andrew L Alexander, Vikas Singh, and Nagesh Adluru. 2018. A natural language interface for dissemination of reproducible biomedical data science. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 197–205.

[304] Jan-Frederik Kassel and Michael Rohs. 2019. Talk to me intelligibly: investigating an answer space to match the user's language in visual analysis. In *Proceedings of the 2019 on Designing Interactive Systems Conference*, 1517–1529.

[305] Jack Cahn. 2017. Chatbot: architecture, design, & development. *University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science*.

[306] Srini Janarthanam. 2017. *Hands-on chatbots and conversational UI development: build chatbots and voice user interfaces with Chatfuel, Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills*. Packt Publishing Ltd.

[307] S Biundo and A Wendemuth. 2016. Companion-Technology for cognitive technical systems. *KI-Künstliche Intelligenz*.

[308] Renaud Blanch and Michel Beaudouin-Lafon. 2006. Programming rich interactions using the hierarchical state machine toolkit. In *Proceedings of the working conference on Advanced visual interfaces*, 51–58.

[309] Abhishek Singh, Karthik Ramasubramanian, and Shrey Shivam. 2019. Introduction to microsoft bot, rasa, and google dialogflow. In *Building an Enterprise Chatbot*. Springer, 281–302.

[310] Marcos Baez, Florian Daniel, and Fabio Casati. 2019. Conversational web interaction: proposal of a Dialog-Based natural language interaction paradigm for the web. In *International Workshop on Chatbot Research and Design*. Springer International Publishing, 94–110.

[311] Donya Rooein, Devis Bianchini, Francesco Leotta, Massimo Mecella, Paolo Paolini, and Barbara Pernici. 2022. Achat-wf: generating conversational agents for teaching business process models. *Software and Systems Modeling*, 1–24.

[312] Artur Caetano, António Rito Silva, and José Tribolet. 2010. Business process decomposition-an approach based on the principle of separation of concerns. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, 5, 1, 44–57.

[313] Jean-Baptiste Louvet, Guillaume Dubuisson Duplessis, Nathalie Chaignaud, Laurent Vercouter, and Jean-Philippe Kotowicz. 2017. Modeling a collaborative task with social commitments. *Procedia computer science*, 112, 377–386.

[314] Anirudh Sundar and Larry Heck. 2022. Multimodal conversational ai: a survey of datasets and approaches. *arXiv preprint arXiv:2205.06907*.

[315] Joao Luis Zeni Montenegro, Cristiano André da Costa, and Rodrigo da Rosa Righi. 2019. Survey of conversational agents in health. *Expert Systems with Applications*, 129, 56–67.

[316] Bogdan Pătruţ and Roxana-Petronela Spatariu. 2016. Implementation of artificial emotions and moods in a pedagogical agent. In *Emotions, technology, design, and learning*. Elsevier, 63–86.

[317] Sherry Ruan, Liwei Jiang, Justin Xu, Bryce Joe-Kun Tham, Zhengneng Qiu, Yeshuang Zhu, Elizabeth L Murnane, Emma Brunskill, and James A Landay. 2019. QuizBot: a dialogue-based adaptive learning system for factual knowledge. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, (May 2019), 1–13.

[318] Michelene TH Chi and Ruth Wylie. 2014. The icap framework: linking cognitive engagement to active learning outcomes. *Educational psychologist*, 49, 4, 219–243.

[319] Luke K Fryer, Kaori Nakao, and Andrew Thompson. 2019. Chatbot learning partners: connecting learning experiences, interest and competence. *Computers in Human Behavior*, 93, 279–289.

[320] Thiemo Wambsganss. 2021. Designing adaptive argumentation learning systems based on artificial intelligence. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–5.

[321] Juan C Castro-Alonso, Rachel M Wong, Olusola O Adesope, and Fred Paas. 2021. Effectiveness of multimedia pedagogical agents predicted by diverse theories: a meta-analysis. *Educational Psychology Review*, 33, 3, 989–1015.

[322] TJ Kennedy and MRL Odell. 2014. Engaging students in stem education. *Science Education International*, 25, 3, 246–258.

[323] Diana Marín. 2021. A review of the practical applications of pedagogic conversational agents to be used in school and university classrooms. *Digital*, 1, (January 2021), 18–33. DOI: `10.3390/digital1010002`.

[324] Pietro Crovari, Chiara Marzano, Massimiliano Nigro, Mariagiovanna Di Iorio, and Franca Garzotto. 2023. Exploring multimodal pedagogical conversational agents for stem education. In *Interaction Design and Children*, Under Review.

[325] Eleni Adamopoulou and Lefteris Moussiades. 2020. Chatbots: history, technology, and applications. *Machine Learning with Applications*, 2, 100006. ISSN: 2666-8270. DOI: `https://doi.org/10.1016/j.mlwa.2020.100006`. `https://www.sciencedirect.com/science/article/pii/S2666827020300062`.

[326] W. Johnson, J. Rickel, and J. Lester. 2000. Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. *International Journal of Artificial Intelligence in Education*, 11, (January 2000), 47.

[327] Fabio Clarizia, Francesco Colace, Marco Lombardi, Francesco Pascale, and Domenico Santaniello. 2018. Chatbot: an education support system for student. In *Cyberspace Safety and Security*. Arcangelo Castiglione, Florin Pop, Massimo Ficco, and Francesco Palmieri, editors. Springer International Publishing, Cham, 291–302. ISBN: 978-3-030-01689-0.

[328] James C. Lester, Sharolyn A. Converse, Susan E. Kahler, S. Todd Barlow, Brian A. Stone, and Ravinder S. Bhogal. 1997. The persona effect: affective impact of animated pedagogical agents. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems* (CHI '97). Association for Computing Machinery, New York, NY, USA, (March 1997), 359–366. ISBN: 978-0-89791-802-2. DOI: `10.1145/258549.258797`. Retrieved 09/02/2021 from `https://doi.org/10.1145/258549.258797`.

[329] Nick Yee and Jeremy Bailenson. 2007. The Proteus Effect: The Effect of Transformed Self-Representation on Behavior. en. *Human Communication Research*, 33, 3, (July 2007), 271–290. ISSN: 0360-3989, 1468-2958. DOI: `10.1111/j.1468-2958.2007.00299.x`. Retrieved 09/02/2021 from `https://academic.oup.com/hcr/article/33/3/271-290/4210718`.

[330] Yanghee Kim, Amy L Baylor, PALS Group, et al. 2006. Pedagogical agents as learning companions: the role of agent competency and type of interaction. *Educational technology research and development*, 54, 3, 223–243.

[331] Rainer Winkler, Sebastian Hobert, Antti Salovaara, Matthias Söllner, and Jan Marco Leimeister. 2020. Sara, the Lecturer: Improving Learning in Online Education with a Scaffolding-Based Conversational Agent. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, (April 2020), 1–14. ISBN: 978-1-4503-6708-0. Retrieved 09/02/2021 from https://doi.org/10.1145/3313831.3376781.

[332] Thiemo Wambsganss, Tobias Kueng, Matthias Soellner, and Jan Marco Leimeister. 2021. ArgueTutor: An Adaptive Dialog-Based Learning System for Argumentation Skills. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (CHI '21). Association for Computing Machinery, New York, NY, USA, (May 2021), 1–13. ISBN: 978-1-4503-8096-6. DOI: 10.1145/3411764.3445781. Retrieved 09/02/2021 from https://doi.org/10.1145/3411764.3445781.

[333] Silvia Tamayo-Moreno and Diana Pérez-Marín. 2017. Designing and Evaluating Pedagogic Conversational Agents to Teach Children. en. 11, 3, 6.

[334] Magnus Haake and Agneta Gulz. 2009. A look at the roles of look & roles in embodied pedagogical agents - a user preference perspective. *I. J. Artificial Intelligence in Education*, 19, (January 2009), 39–71.

[335] Kristen Blair, Daniel L Schwartz, Gautam Biswas, and Krittaya Leelawong. 2007. Pedagogical agents for learning by teaching: teachable agents. *Educational Technology*, 56–61.

[336] Catherine C. Chase, Doris B. Chin, Marily A. Oppezzo, and Daniel L. Schwartz. 2009. Teachable Agents and the Protégé Effect: Increasing the Effort Towards Learning. en. *Journal of Science Education and Technology*, 18, 4, (August 2009), 334–352. ISSN: 1573-1839. DOI: 10.1007/s10956-009-9180-4. Retrieved 09/02/2021 from https://doi.org/10.1007/s10956-009-9180-4.

[337] Jessy Ceha, Ken Jen Lee, Elizabeth Nilsen, Joslin Goh, and Edith Law. 2021. Can a Humorous Conversational Agent Enhance Learning Experience and Outcomes? In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Number 685. Association for Computing Machinery, New York, NY, USA, (May 2021), 1–14. ISBN: 978-1-4503-8096-6. Retrieved 09/02/2021 from https://doi.org/10.1145/3411764.3445068.

[338] Yuichiro Anzai and Herbert A Simon. 1979. The theory of learning by doing. *Psychological review*, 86, 2, 124.

[339] Yanghee Kim, Amy L Baylor, and Entong Shen. 2007. Pedagogical agents as learning companions: the impact of agent emotion and gender. *Journal of Computer Assisted Learning*, 23, 3, 220–234.

[340] José Miguel Ocaña, Elizabeth K. Morales-Urrutia, Diana Pérez-Marín, and Celeste Pizarro. 2020. Can a learning companion be used to continue teaching programming to children even during the covid-19 pandemic? *IEEE Access*, 8, 157840–157861. DOI: 10.1109/ACCESS.2020.3020007.

[341] Jay Cross. 2011. *Informal learning: Rediscovering the natural pathways that inspire innovation and performance*. John Wiley & Sons.

[342] Florian Weber, Thiemo Wambsganss, Dominic Rüttimann, and Matthias Söllner. 2021. Pedagogical agents for interactive learning: a taxonomy of conversational agents in education. In *Forty-Second International Conference on Information Systems, Austin*.

[343] Wilbert O Galitz. 2007. *The essential guide to user interface design: an introduction to GUI design principles and techniques*. John Wiley & Sons.

[344] Lucy Green. 2008. Group cooperation, inclusion and disaffected pupils: some responses to informal learning in the music classroom. presented at the rime conference 2007, exeter, uk. *Music Education Research*, 10, 2, 177–192.

[345] Mina C Johnson-Glenberg. 2019. The necessary nine: design principles for embodied vr and active stem education. In *Learning in a digital world*. Springer, 83–112.

[346] Mina C Johnson-Glenberg, David A Birchfield, Colleen Megowan-Romanowicz, and Erica L Snow. 2015. If the gear fits, spin it!: embodied education and in-game assessments. *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS)*, 7, 4, 40–65.

[347] Heather L O'Brien, Paul Cairns, and Mark Hall. 2018. A practical approach to measuring user engagement with the refined user engagement scale (ues) and new ues short form. *International Journal of Human-Computer Studies*, 112, 28–39.

[348] Deanna M Spanjers, Matthew K Burns, and Angela R Wagner. 2008. Systematic direct observation of time on task as a measure of student engagement. *Assessment for effective intervention: official journal of the Council for Educational Diagnostic Services*, 33, 2, (March 2008), 120–126. ISSN: 1534-5084. DOI: 10.1177/1534508407311407.

[349] Jennifer A Fredricks, Phyllis C Blumenfeld, and Alison H Paris. 2004. School engagement: potential of the concept, state of the evidence. *Review of educational research*, 74, 1, (March 2004), 59–109. ISSN: 0034-6543. DOI: 10.3102/00346543074001059.

[350] Ryan Schaaf. 2012. DOES DIGITAL GAME-BASED LEARNING IMPROVE STUDENT TIME-ON-TASK BEHAVIOR AND ENGAGEMENT IN COMPARISON TO ALTERNATIVE INSTRUCTIONAL STRATEGIES? en. *The Canadian Journal of Action Research*, 13, 1, 50–64. ISSN: 1925-7147, 1925-7147. DOI: 10.33524/cjar.v13i1.30.

[351] Leo Natan Paschoal, Aliane Loureiro Krassmann, Felipe Becker Nunes, Myke Morais de Oliveira, Magda Bercht, Ellen Francine Barbosa, and Simone do Rocio Senger de Souza. 2020. A systematic identification of pedagogical conversational agents. In *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[352] Marie-Luce Bourguet. 2006. Towards a taxonomy of error-handling strategies in recognition-based multi-modal human–computer interfaces. *Signal Processing*, 86, 12, 3625–3643.

[353] Janienke Sturm and Lou Boves. 2005. Effective error recovery strategies for multimodal form-filling applications. *Speech communication*, 45, 3, 289–303.

[354] Diana Pérez-Marín. 2021. A review of the practical applications of pedagogic conversational agents to be used in school and university classrooms. *Digital*, 1, 1, 18–33.

[355] Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio, and Alfonso Pierantonio. 2020. Supporting the understanding and comparison of low-code development platforms. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 171–178.

[356] William Villegas-Ch, Joselin García-Ortiz, and Santiago Sánchez-Viteri. 2021. Identification of the factors that influence university learning with low-code/no-code artificial intelligence techniques. *Electronics*, 10, 10, 1192.

[357] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. End-User Development: An Emerging Paradigm. en. In *End User Development*. Human-Computer Interaction Series. Henry Lieberman, Fabio Paternò, and Volker Wulf, editors. Springer Netherlands, Dordrecht, 1–8. ISBN: 978-1-4020-5386-3. DOI: `10.1007/1-4020-5386-X_1`. Retrieved 11/21/2022 from `https://doi.org/10.1007/1-4020-5386-X_1`.

[358] Matt McCutchen, Shachar Itzhaky, and Daniel Jackson. 2016. Object spreadsheets: a new computational model for end-user development of data-centric web applications. In *Proceedings of the 2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, 112–127.

[359] Simon Peyton Jones, Alan Blackwell, and Margaret Burnett. 2003. A user-centred approach to functions in Excel. *ACM SIGPLAN Notices*, 38, 9, (August 2003), 165–176. ISSN: 0362-1340. DOI: `10.1145/944746.944721`. Retrieved 11/26/2022 from `https://doi.org/10.1145/944746.944721`.

[360] G. Fischer, E. Giaccardi, Y. Ye, A. G. Sutcliffe, and N. Mehandjiev. 2004. Meta-design: a manifesto for end-user development. *Communications of the ACM*, 47, 9, (September 2004), 33–37. ISSN: 0001-0782. DOI: `10.1145/1015864.1015884`. Retrieved 11/03/2022 from `https://doi.org/10.1145/1015864.1015884`.

[361] Timothy C. Lethbridge. 2021. Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering. en. In *Leveraging Applications of Formal Methods, Verification and Validation* (Lecture Notes in Computer Science). Tiziana Margaria and Bernhard Steffen, editors. Springer International Publishing, Cham, 202–212. ISBN: 978-3-030-89159-6. DOI: `10.1007/978-3-030-89159-6_14`.

[362] Gwendal Daniel, Jordi Cabot, Laurent Deruelle, and Mustapha Derras. 2020. Xatkit: a multimodal low-code chatbot development framework. *IEEE Access*, 8, 15332–15346.

[363] Samuel M Scheiner and Jessica Gurevitch. 2001. *Design and analysis of ecological experiments*. Oxford University Press.

[364] Asbjørn Følstad, Theo Araujo, Effie Lai-Chong Law, Petter Bae Brandtzaeg, Symeon Papadopoulos, Lea Reis, Marcos Baez, Guy Laban, Patrick McAllister, Carolin Ischen, et al. 2021. Future directions for chatbot research: an interdisciplinary research agenda. *Computing*, 103, 12, 2915–2942.

[365] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, 102274.

[366] Yiqiu Shen, Laura Heacock, Jonathan Elias, Keith D Hentel, Beatriu Reig, George Shih, and Linda Moy. 2023. Chatgpt and other large language models are double-edged swords. (2023).