POLITECNICO MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

An architecture to implement emphatic dialogue generation in a robot interacting with a human in improvisational settings

Tesi di Laurea Magistrale in
Computer Science Engineering - Ingegneria Informatica

Author: **Simone Cattaneo**

Student ID: 970296
Advisor: Prof. Andrea Bonarini
Academic Year: 2022-23

Abstract

The goal of social robotics is to build robots able to interact with humans by selecting proper reactions coherent with the context and with the emotional state manifested by the human agent. Improvisational theatre is the perfect environment where the social behaviors can be tested in an unconstrained setting since the stage is a space outside the rules of time and space, customizable and in which any kind of situation can be created.

The goal of this project is to implement on a robot the ability to recognize the verbal stimuli from a interlocutor, extract the emotions inherently contained in the detected sentences, generate a suitable response, and reproduce it through speakers, in order to create a turn-taking conversation. Since the humans perceive in a more positive way the relationships with interlocutors that are able to properly perceive and manifest emotions, the model with the purpose of generating the robot verbal responses is trained to show empathy and awareness of the emotional state of the current situation. Finally, the only strict constraint is to keep everything locally on the disk of the used embedded system to be completely independent from the possible breaks that the use of online APIs would introduce.

Keywords: Autonomous Robot, Improvisation, Language, Empathy, Human-Robot interaction, Language model, ROS

Abstract in lingua italiana

Lo scopo della robotica sociale è di costruire robot in grado di interagire con esseri umani scegliendo le migliori reazioni coerenti con il contesto e con lo stato emozionale manifestato dall'agente umano. Il teatro di improvvisazione è il contesto perfetto in cui i comportamenti sociali possono essere testati in un ambiente non vincolato in quanto il palco è uno spazio al di fuori dello spazio e del tempo, personalizzabile e in cui può essere ricreata qualsiasi situazione.

L'obiettivo di questo progetto è quello di implementare su un robot la capacità di riconoscere gli stimoli verbali provenienti da un interlocutore, di estrarre le emozioni intrinsecamente contenute nelle frasi identificate, di generare una risposta adeguata e di riprodurla attraverso gli altoparlanti, in modo tale da creare una conversazione a turni. Siccome gli esseri umani percepiscono in modo più positivo le relazioni con interlocutori che sono in grado di percepire e manifestare adeguatamente emozioni, il modello con lo scopo di generare le risposte verbali è allenato per mostrare empatia e consapevolezza dello stato emotivo della situazione corrente. Infine, l'unico vincolo è quello di mantenere tutto in locale sul disco del dispositivo utilizzato, in modo da essere completamente indipendente dai possibili guasti che l'uso di API in rete potrebbero introdurre.

Parole chiave: Robot autonomo, Improvvisazione, Linguaggio, Empatia, Interazione uomo-robot, Modello di linguaggio, ROS

Contents

1 | Introduction

Imagine to go back in time about 70 years. In 1946 was born Eniac, the very first fully electronic computer built by the americans J.Mauchy and J.Eckbert: it weighed around 30 tons, it needed a 150 square metres room to be placed and used thousands of valves [6]. In the first years of the 50s the first commercial computer called UNIVAC was created, and its value was around 1 million dollars [80]. Now imagine that in the 1950 a paper begins with a question: "Can a machine think?", proposing the vision of a computer able to act like a human being. In this famous paper by Alan Turing the author introduced the "Imitation game" (also known nowadays as "Turing test") to determine the intelligence of a machine. Very briefly, a computer pass it if is able to act so similarly to a human being to "fool" a human interlocutor, making him believe that is not interacting with a machine [77].

Obviously, in the world in which the paper was published this vision maybe seemed an utopia, but several decades after it my be not. In the last years the growing of machine learning and deep learning research is incredible, and has led to the deployment of models that can perform with great accuracy various tasks. Now we have models able to perform visual recognition, models that make it possible to have self-driving cars, models that can be applied in healthcare saving lives, models that can create an image from scratch starting from a description, and almost everything we can just imagine. And finally models that deal with natural language. This kind of models are in the spotlight in these last months thanks to Chat-GPT: the ability of this model to answer all our questions in almost any natural language is impressive, and maybe for some people unsettling.

But, forgetting for a moment the infinite possible applications of Chat-GPT, our doubt is now if these models can interact with us in a "human" way, making us forget that we are dealing with a machine. The perfect context in which the conversational abilities can be tested is the improvisational theatre [55]. The stage becomes an isolated space in which the actors can collaborate in order to create stories, situations and dialogues, often strange and funny due to the lack of a written and prepared script. And if one of the actors is replaced by a machine, maybe equipped with a robotic body that allows it to act both physically and verbally, its flexibility, adaptability and ability to interact with a human can effectively be tested.

1.1. Description of the work

This thesis work is part of a larger project whose aim is to develop a robot able to interact with a human person in the specific context of improvisational theatre. The desired goal would be to make the robot able to detect the emotion expressed by the human actor through its movements and words, and react in the proper way expressing, both physically and verbally, an adequate emotion.

Several studies and previous thesis have brought the designed robot to the actual situation. In particular, dr. Julian Fernandez and prof. Andrea Bonarini studied how emotional states can be represented exploiting particular features such as the movements and how human perceive them. They concluded that emotions of a non-anthropomorphic robot can be recognized by a human observer, thank to modulation of linear and angular velocities and to the placement in a context [29–31].

This work was then expanded in the graduation thesis of L. Bonetti under the supervision of prof A. Bonarini. He implemented on an already existing robot the ability of acting a pre-defined script, written in a particular language that specifies the actions it has to perform. The robot is then programmed in order to follow the indications of the "script", locating itself on the stage and understanding the right moment to perform the actions. We can say the robot was able to synchronize itself to the human actor in order to perform brief "scenes", but it wasn't able to react to his/her actions [33].

Then, with the thesis works of L. Farinelli and C. Chiroli, always supervised by prof. A. Bonarini, the ability to act in an improvisational context was implemented. In particular, the action of the human actor is classified in a discrete set of 16 scenic actions basing on [46]:

- the emotion, expressed by both the face and the body pose;
- the proxemic zone, which is the zone occupied by the actor with respect to the robot and determine the intimacy he has with it;
- the proxemic movement, determined by the direction and speed of the movement of the actor.

The robot is then able to select the proper reaction basing on a Finite State Automaton that accounts for the personality of the robot (defined in advance), the last scenic action of the human actor and the current emotional state of the robot [36].

It is clear at this point that the robot is able to interact with a human but it completely lacks the verbal dimension; this is the contribution of my research. The goal I wanted to achieve was to implement the chance to have a conversation with the robot in an improvisational context. To be more specific:

- the robot should be able to be an empathetic interlocutor in a dialogue, giving coherent answers and expressing emotions adequate for the context;
- the robot should be able to pronounce sentences coherent with the scenic action performed by the human actor, as detected by the already existing module;
- the robot should be able to synchronize with the human interlocutor, waiting that he/she finishes to pronounce his/her sentence and/or perform his/her action before generating an answer, possibly introducing the smallest possible latency.

It was clear from the beginning that the proper technology to generate answers in a conversational context was a generative language model, but since it was a very new topic in this line of research inside the laboratory the main issue I had to face was to identify the most appropriate model and dataset to train it. The only big constraint was to keep any computation local without using any API to connect to online services, in order to be independent from the internet connection and avoid latency introduced by breakdowns and slowdowns.

The final result is quite satisfying: I developed a module able to verbally interact in a conversational context with a human agent by detecting his speech, generating an emphatic response (which is most of the time coherent), extracting the emotion inherently contained in the interlocutor sentence and producing the audio of the response. The main critical point that should be improved in a future work is the synchronization and the integration of the two big modules (language and movement).

1.2. Structure of the thesis

The structure of the document is the following:

- Chapter 1 (this chapter) contains an introduction of the thesis work, specifying the context, the starting point, the desired and reached goal;
- Chapter 2 define the theoretical background needed to understand the content of the work;

- Chapter 3 describes the approach I have followed to reach the goal, mainly focusing on the technologies I have used;
- Chapter 4 contains a precise description of the software architecture I have developed;
- Chapter 5 describes both the quantitative and qualitative results obtained;
- Chapter 6 summarizes the conclusions reached and introduces some possible future developments.

2 | Theoretical background

In this chapter, several theoretical concepts, fundamental to fully understand the topics and the context around them, are presented and explained. In particular the following presentation will be divided into two main parts:

- first of all it has to be clarified what is the context in which the final system will be tested, which is the improvisational theatre, as well as several notions such as what is meant with language, emotion, empathy and why are they important also from a social perspective for the inter-personal interactions (and more specifically during a dialogue);
- then few examples of results reached by the research that are relevant to better understand why the above concepts are important in the field of human-robot interaction will be exposed.

2.1. Theatre and improvisation

Theatre is one of the oldest form of art, used far from the ancient Greeks to entertain an audience creating a representation of real or invented events through the collaborative performance of the actors. It can be defined as the art of "mimesis" or of "make believe", since it is not important that the represented act is realistic, but it is fundamental that the actors involved in the scene make audience "believe" in what they are doing and what they are feeling, through their gestures, their speech, their facial expressions [26, 35]. We can say that what happens on the stage, which is the designed space of the performance[79], should not be real in the general meaning of the word, but it should be inside the specific context of the stage itself.

While traditional theatre is generally characterized by the presence of a written script followed by the actors, the improvisation is a form of live theatre in which everything is created at the moment without a predefined line: the plot, the characters and the dialogues [83]. It follows that there will never be an improvisational performance equal to another one [27]. There are two main kinds of improvisation [14]:

- Short-form improv: short and self-contained scenes. This is the type we are most interested in.
- Long-form improv: series of interlocked scenes, with callbacks, recurring characters and overarching plot. This is obviously much more complex than the short one and requires building a plot in real time.

We can say that the main features of the improvisation are the spontaneity and the ability of the actors to quickly come up with the next activity in order to move the plot forward.

From now on I will consider only short-form improvisation with two actors, since it is the framework used in this work. Maybe now it is clearer why this is the perfect environment in which the research outlined in the introduction can be developed and tested:

- The stage is a physical area but outside the space and time rules of the external world. Here, anything can happen, there are no strict constraints, everything is fake but the relationships and the emotions of the characters created by the actors.
- In an improvisation there are not rigorous rules and evident errors, since every action performed by an actor, strange as it may be, can be an opportunity for the other actor to introduce new plot elements or to react in a fancy and funny way. This process of integrating new emerging subjects in the previous circumstances is called *justification* [34];
- By simplifying, we can reduce the improvisation to a sequence of couples action-reaction. The most relevant thing is that the second one is coherent from a conceptual and emotional point of view to the first one and that the latency between them is not too consistent.
- Finally, theatre is basically a matters of humans. It is made of human beings that by expressing emotions through their voices and gestures try to make other human beings (the audience) perceive their feelings. It is apparently so distant from the cold and inexpressive material of the machines that can be the perfect field to test the social interactions between robots and humans.

2.2. Language

Following a top-down approach, after having presented the environment in which the system should be developed, we have now reached the real protagonist of this work: language. First of all, language is a structured system of communication composed by a vocabulary, which is a set of familiar words, and by a syntax, which can be defined as a set of constraints on how the elements of the vocabulary can be used in order to create more complex structures (such as clauses and sentences). The other core concept of language is semantics, which is the meaning that is bounded to a single word, an expression or an entire sentence [44].

2.2.1. Language and evolution

Since the goal of this thesis is to create a system able to replicate as closely as possible the human ability to speech, it may be interesting to analyze the origin of human language and how it evolved through time. It is from an evolutionary point of view an open problem: even if lots of animal species have developed some kinds of communication processes only few of them can be considered to have an effective language system, and no one as complex and sophisticated as humans. There are several hypothesis to explain the reasons of the human language evolution. For example, a reason could be to facilitate the exchange of factual information, or for "idle chit-chat", or, maybe, for some more useful purpose.

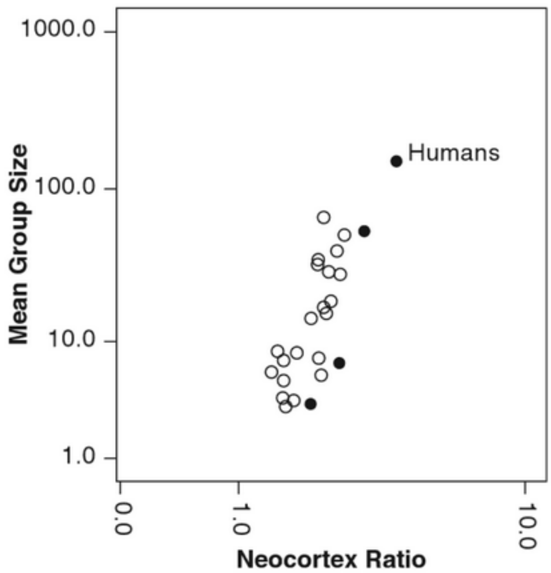


Figure 2.1: Mean social group size plotted against neocortex ratio for individual primate genera. Humanoids (solid circles) are distinguished from monkeys.

A very interesting theory from [41] says that language evolved because a better system than social grooming was needed in primates as bonding mechanism. As we can see in figure 2.1, hand in hand with the evolution of the species an increase in the size of the brain is observed together with the dimension of the social groups (this is the social brain hypothesis, according to which primates evolved larger brain to manage their complex social system [42]): this led to the need of having a more efficient social mechanism, such as the language.

2.2.2. Language complexity

What is clear is that this is not an easy ground to move on. It is interesting to think how easily we manage this complex matter in the everyday life, forgetting that it is not trivial at all.

Language relies on social conventions. For a baby a very simple word like "pen" is just a sound that can be imitated, but without any meaning associated. It is through the experience that a meaning is bounded to the sound, and the word "pen" can be used to design a concrete object in the word. This is a very simple example, but when a sound can't be directly mapped to a concrete object that can be touched in the reality (such as abstract concepts like "fantasy" or "fear") the processes to associate a meaning to a sound are much more complex [51]).

Language is ambiguous and vague. Linguistic vagueness means that it could be difficult to assign a meaning to an expression or a sentence. On the other hand, linguistic ambiguity is a feature of language that means that a sentence or an expression can be interpreted in multiple ways, with the consequence that the interpretation for another human being (and obviously also for an AI system) is difficult or even impossible. There are two kinds of ambiguity:

- lexical ambiguity: a word or sentence can have multiple meanings;
- structural ambiguity: arises when there could be different interpretations according to the different order of words or phrases;

Another element that may lead to ambiguity is the presence of figurative languages, that include figures of speech such as metaphor, use of irony, idioms, puns, images and sound-based devices. The use of these can lead to problems of comprehension for non-native speakers and NLP (natural language processing) systems [18].

Language is variegated. The number of existing languages in the world is estimated to be more than 5000 [44]. Some of them can be spoken, other signed or both, and for each of them differences can exist depending on geographic location and social reasons. This implies that even if people share the same mother language, a difference in the geographical region where they leave can bring to a difference in their accent and linguistic inflection and even terms, generating difficulties in the mutual comprehension.

2.2.3. Language and social interaction

Since we will to develop and test a system to be able to interact with a human, it is interesting to briefly clarify what is meant with social interaction and the role of language in the inter-personal relationships, as exposed in [47]. First of all, something is defined as "social" when it is done with or in relation to other people, so it is clear that at least two persons should be involved. Generally, an interaction is "intrinsically socially structured", which means that what is done or more specifically said by person A towards person B, influences the response that can be given by person B.

Example 1
A: Hi, my name's Brian.
B: Hi, I'm John.

Example 2
A: Excuse me?
B: Yes?

Example 3
A: [*wringing his hand*] SHIT!
B: Are you OK?

Figure 2.2: Example of interactions between 2 people.

In every example of 2.2 there is a link between the two utterances: the actions of the people involved are tied, because as Francis and Hester [47] stated

> an action project the kind of thing that can or should be done next.

While the sociability and the tendency to aggregate in groups is what characterize humans and has elevated us above primates (as explained in section **??**), language is the main mean by which the social activity is realized. It lets us frivolously chit-chat with other people, make questions, answer, argument, debate, argue, describe situations, but also it makes possible the existence of institutional structures. Communication is the necessary condition of social life, as well as it is, probably, what makes us humans.

2.3. Emotions

2.3.1. What is an emotion?

Before understanding why emotions are fundamental in human interaction a general description of what are them should be given. The problem is that there is no consensus on a unique and precise definition. This is because, first of all, it is a very complex subject also related to the concepts of consciousness and mind which have challenged the best scientists and philosophers since the ancient times, but it is also due to the cultural difference in their conception and interpretation, as well as to the ambiguity of the term itself. Indeed, there are constructs that are often used as synonyms such as feelings, affects and mood, that have different shades of meaning [56].

Anyway, just to give a very high level overview, few interesting definitions extracted from the huge pile of the existing ones will be enough to give a general idea of what we are talking about. Starting from the most ancient one from Aristotle

> Emotions are all those feelings that so change men as to affect their judgements, and that are also attended by pain or pleasure. Such are anger, pity, fear and the like, with their opposites.

Or without going so far in time, an emotion " is a complex behavioral phenomenon involving many levels of neural and chemical integration" that involves nervous and hormone mechanisms [62], or even a "complex experience of consciousness, bodily sensation, and behaviour that reflects the personal significance of a thing, an event, or a state of affairs." [5].

Emotion is a complex behavioral phenomenon involving many levels of neural and chemical integration. In this review the author deals with a variety of physiological changes and discusses the nervous and hormone mechanisms involved in emotional integration

2.3.2. Emotions and social interactions

What is clear is that this a very abstract and complex subject, hard to fix with words but that influences our everyday behavior and our relationships with other people.

An emotion can be decomposed into 3 main components [4]:

- Subjective component: how a person perceives an emotion;
- Physiological component: how the body reacts to an emotion;

- Expressive component: how the person behaves feeling that emotion.

These components are very personal since the way a person perceives, processes and manifests an emotion is sensitive to subjectivity, and it can influence the emotion and consequently the possible reactions of other people. This is also related to the so-called emotional intelligence [65], which is the ability to control the own emotions to choose the best behavior in a given situation, also considering the emotional states of the interlocutors. More precisely, four branches, that define four areas of skills, of emotional intelligence can be identified [65].

- Perceiving emotion. It involves the reception of non-verbal expression of emotion, that come from facial expressions and body language. The ability of identifying in an accurate way the emotions from the face or voice is a crucial skill for emotional intelligence.
- Using emotions to facilitate thoughts. This is the ability to use emotions in order to "think better": when something is considered important and grabs the attention it generates an emotional reaction, then emotions can be used to direct the thinking effort towards really important matters.
- Understanding emotions. Each emotion brings important information, for example the thoughts of the person who manifest it or the possible actions that could follow. Understanding these implicit messages means also to understand and reason about the meaning of the emotions themselves, and this is an important skill for emotional intelligence;
- Managing emotions. This is the most important skill, since involves the ability to regulate and manage the own and most of all the others' emotions in order to achieve a social goal.

Indeed, an interesting research in this field is about the power of emotion regulation, which means that a person may be able through the way he behaves to regulate and even modify the emotional response of other people. In this case, we speak about social regulation, contrasted to the self-regulation (see figure 2.3) necessary to maintain a mental and physical wellness. It obviously has important consequences in the ways feelings and emotions, that can be expressed more or less explicitly through the spoken language but even more through gestures, body movements and face expressions, can affect social interactions [70].

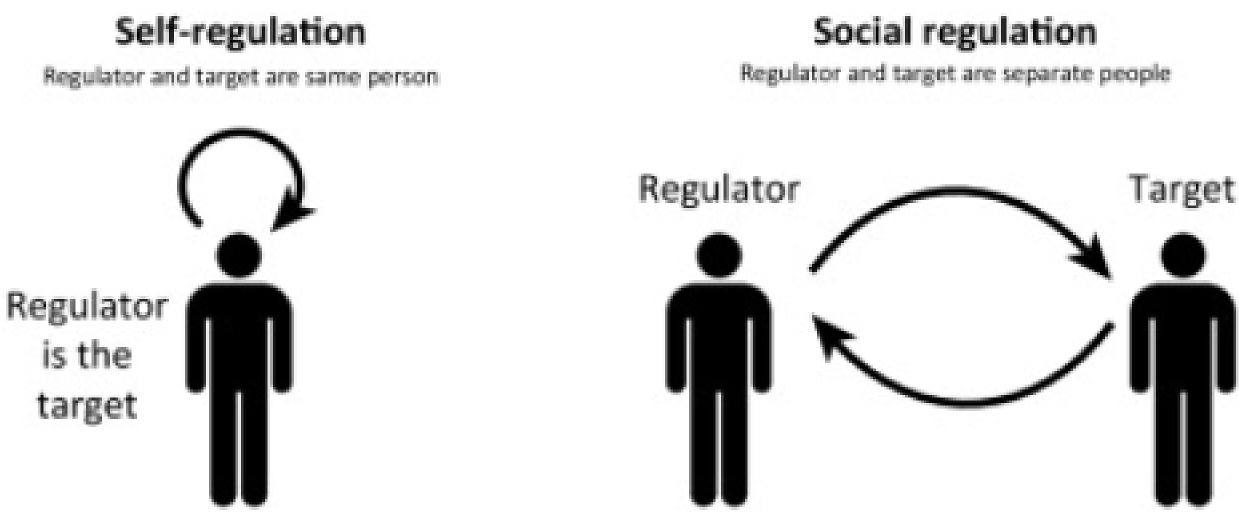


Figure 2.3: Self regulation and social regulation.

2.3.3. Emotion model

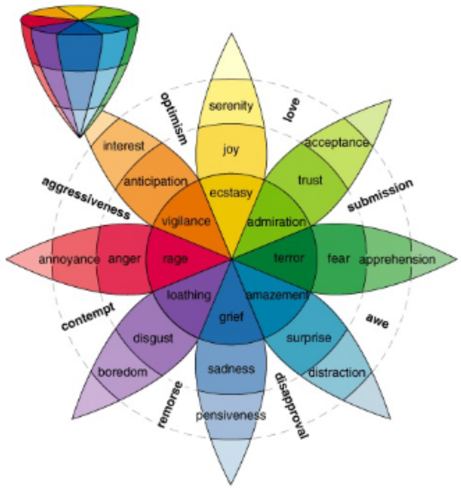
As mentioned above, emotions are complex, subjective and non-deterministic, since the same event (or in general external stimulus) can generate different emotional states in different individuals. In order to better study emotions and their consequences is necessary to introduce a computational model to define a classification. The computational model used for this project was chosen in a previous contribution by L. Farinelli [46], but it is useful to briefly expose some of the existing models as well as the selected one, since this choice had some consequences in the development of the present work.

- Circumplex model of affect. It was introduced by James Russel in [72]; the emotions are mapped in a Cartesian reference system as a linear combination of the concepts of *arousal* and *valence*.
- Plutchik's model. It was presented by Robert Plutchik in [67] and considers the existence of eight primary emotions from which all the others can be derived. The emotions can be then organized in concentric circles where the inner ones are more basics and the outer are more complex, generating a "wheel" structure.
- Tomkin nine affects theory. It was developed by Silvan Tomkin [75, 76] and considers the existence of nine primary affects classified in positive, negative and neutral. Affects are defined as innate and evolutionary processes that amplify triggering information and are different from emotions, which are complex combinations of affect and personal memories.
- Ekman model. This was introduced by Paul Ekman [43] and it is also known as

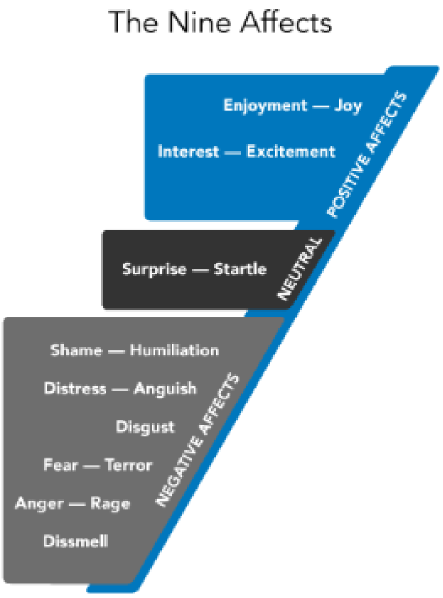
"discrete model" since it theorizes the existence of six basic emotions that are shared among all the humans regardless their culture and geographical position, that are: anger, disgust, fear, joy sadness and surprise. The word "discrete" is referred both to the classification of the basic emotions and to the facial patterns from which the emotions themselves can be deduced.

The model finally selected was the Ekman one. The main reason was that, since the goal was to build a system able to identify the emotional state of a human interlocutor from his non-verbal behavior and compute a suitable reaction, a discrete model with limited number of possible outcome emotions was much easier to manage. Tis same reason is making the Ekman's model also the most used in robotics and affective computing systems. This choice influenced my work since I had to adapt the sentiment analysis module, which extracts an emotion from the interlocutor's sentence, to map the result in one of the six Ekman emotions, as will be explained later.

(a)



(b)



(c)

Figure 2.4: (a) Circumplex model of affect. (b) Plutchik's model. (c) Tomkin's model

2.4. Empathy

The origin of the word empathy comes from the German term "einfuhlung" which literally means "in-feeling" and was used by the psychologist Theodore Lipps in 1880s to describe the appreciation from a emotional point of view of another's feelings [54]. Later on the notion of empathy was expanded and became a complex and ambiguous subject often confused with other associated concepts such as compathy (share someone feelings due to shared situations), mimpathy (imitation of someone emotion without experiencing it), sympathy (an immediate and uncontrolled emotional reaction that arises when someone imagines himself in the situation of someone else), transpathy ("emotional contagion", which means being influenced and "infected" by someone else's emotional state) and unipathy (intense form of transpathy).

2.4.1. What is empathy?

But then, what really is empathy? The concept can be trivialized as the ability to feel what an interlocutor is feeling assuming his perspective with respect to a situation [**?**], and it is also related to the notion of emotional intelligence introduced in 2.3.2. Anyway, it is not so easy to fix empathy in a single definition (and the proof of this are the tens of different related statements that can be found) since some conceptual problems that make the matter quite ambiguous should be clarified. These arguments from [37] are summarized below.

- Empathy includes both cognitive and affective dimensions: the first is the ability to understand the others' emotional state, the second is about effectively experiencing an emotion in response to emotional stimuli.
- Congruency problem: emotion perceived by the observer can be very similar to the one felt by the observed, but not necessarily identical.
- Empathy can be fired in response to external stimuli even if the emotion is not directly perceived. For example a person who minimizes an accident or a third party that verbally elicits empathy towards an absent person.
- Empathy is a "trait" because it can be defined as an "ability" and considered a personal inclination, but it also depends on the context, since there are situations in which it could be more easily elicited.
- Empathy has not necessarily a behavioral outcome, even if it is often present. Consider for example competitive situations in which someone has personal interests.
- Empathy is naturally and automatically fired, but can be controlled in several ways (for example not thinking or avoiding a situation).

The paper finally summarizes the empathy concept fixing it in precise and concise statements that I will reuse, defining it as

> *an emotional response (affective), dependent upon the interaction between trait capacities and state influences. Empathic processes are automatically elicited but are also shaped by top-down control processes. The resulting emotion*

is similar to one's perception (directly experienced or imagined) and understanding (cognitive empathy) of the stimulus emotion, with recognition that the source of the emotion is not one's own. [37]

2.4.2. Importance of empathy in social interactions

It is inherently part of the human being to use empathy, often even without being aware, as a social mechanism to be used during interactions with other people to build effectively working relationships and networking. It is also often considered the basis and the mediation for social cooperation and prosocial behavior, which is any action finalized to the improvements of the welfare of another being [38]. Briefly, it can be considered a necessary condition for sociability since it allows to lead pleasant social interactions by choosing proper responses to the other's actions, and maintaining positive relationships with the interlocutors.

It is obvious at this point that in order to develop a robotic system able to interact with a human agent emotional intelligence and empathy (or at least the appearance of them) are features to be implemented. An interesting study in this field is [58], where an autonomous robot is used in order to comment the performances of two humans playing chess. The robot was programmed to show emphatic reactions through facial expressions and verbal comments (these last ones particularly interesting for this thesis) to the moves of one player and neutral reactions to the moves of the other one. Figure 2.5 shows examples of utterances and can be noticed that neutral reactions are limited to comments about the quality of the move (that can be *"good"* or *"bad"*), whereas the emphatic reactions contain references to the feelings of the player and try to motivate him. The final results showed that the player toward which the robot behaved emphatically perceived it as "friendlier", underlying the key role of empathy in human-robot interactions.

2.5. State of the art

As described in 1.1 the goal of this thesis work was to implement on an already existing robot able to interact with a human agent through body movements the ability to interact also on the verbal dimension, creating dialogues conceptually and structurally coherent. I didn't find in the literature meaningful researches with the same context and objective, but there are still several works that are really interesting and that inspird my work. In particular, I found particularly useful projects where generative language models such as GPT-2 and GPT-3 (that will be introduced and explained in detail in next chapters) have

Affective state	Meaning	Examples of verbal utterances	
		Empathic	Neutral
Stronger reward	Better than expected	"Great move! Even better than I was expecting!"	"Good move."
Expected reward	As good as expected	"Nice move, you played what I would have played!"	"You played well this time."
Weaker reward	Not as good as expected	"I believe you could have played better."	
Unexpected reward	Good, unexpected	"You played very well."	
Unexpected punishment	Bad, unexpected	"Oh... I wasn't expecting that move..."	"This move wasn't that good."
Weaker punishment	Not as bad as expected	"Well done, you are recovering your disadvantage!"	
Expected punishment	Bad, as expected	"Don't worry, you didn't have better options."	"Bad move."
Stronger punishment	Worse than expected	"You're making me sad... you could have played better."	

Figure 2.5: Verbal comments of the robot, both emphatic and neutral, for different situations. The utterances were expanded with name of the players at the beginning r at the end.

been used in innovative ways in artistic fields from literature to theatre, as well as the development of chatbots able to emphatically interact with an interlocutor.

2.5.1. THEaiTRE

The aim of the project described in [71] was to use deep learning models to generate a theatre script. The selected language model was GPT-2, which is a 1.5 billion parameters transformer model developed by openAI trained on a dataset containing 8 million web pages. It was used a hierarchical setup, which means to use a language model in order to generate a title, then a synopsis from that title which should include the list of characters involved in the story and finally generate the script. Even if the dataset with which GPT-2 was pre-trained must contains some theatre plays, the output generated by the model when prompted with an input was not satisfying for a drama: it lacked of a plot, there weren't scenes moving towards a conclusion, characters appeared randomly. The ideas to improve the quality of the result were: define a set of predefined characters (contained in the generated synopsis), define the order with which they had to speak, fine-tune a language model for each character in order to make it specialized for the specific personality of its character, add the name of the current character and use the designed language model corresponding to it to generate only his part.

Complementary activities were:

- Data summarization: the used dataset lacked of some information needed for the hierarchical approach, since data generally contains only title and script. The synopsis were generated by exploiting data summarization task on the available scripts.
- Machine translation: it was used to generate outputs both in English and Czech languages.
- Human in the loop: collaboration of machine with human to obtain better results.

2.5.2. Language model as narrator in improvisational theatre

In this project introduced in [34] the improvisational theatre context was used as a test environment for the generative power of language models. The selected model was GPT-3, a huge 175B parameters model developed from openAI that represent the present state of the art (even if chat-GPT that is also called GPT-3.5 has been recently released). In this case, the aim was not to create the part of each character in the script, but to support the human narrator in creating the story of the improvisation and, particularly interesting, everything was done during a live performance. The human operator typed a sentence that was concatenated to the previous context and given as input to GPT-3 to generate three different outputs up to 100 characters length. The operator could then select a number of those suggestions (even none or all) and add them to the context. The result was a combination of the unpredictable generations of the language model and the conscious collaboration of the human operator in 1order to create new but even fancy and funny stories on which actors could improvise, as can be seen in 2.6.

The outcome was satisfying both from the audience that appreciated the contribution of the AI that introduced a level of *craziness* and *randomness* and from the actors, since the model generating the plot removed from them the task of moving the plot forward allowing to focus only on the relationships on the stage and on the engagement of the audience.

At the Pizza Hut. Brian and his date lost patience. (The operator misunderstood the relationship between the two protagonists.)
There was always a reason for them to admire each other. Brian was an expert at making pizza. Sally found her vocation, making pizza like Brian. Brian started listing all the products... Baguettes, patisserie... Sally asked Brian for help. (The operator made a confusion in the name, as it was Sandra, not Sally.)
The door opened and a burly man entered, followed by his wife. (A couple entered the pizzeria, the man spoke with a heavy voice.)
The husband and the wife entered the pizzeria. They asked for supremes, with garlic bread. **Both women had crushes on Brian.** (The unnamed wife briefly approached Brian.)
Sally searched for pastries. **The husband and the wife asked for vodka.** (Unused suggestion.)
They got creme patissiere...
Brian apologized.
(Sally/Sandra was rolling pizza on the floor.)
Sally was dreaming about becoming a master patissier. **She continued to look for pastries.** (Sally/Sandra said she was done working at Pizza Hut and wanted to resign. Scene transition, with an angry boss entering the stage.)
Brian's boss told him he would let her go. Sally gave her notice. The boss refused. The boss was cruel. **Brian asked the boss for her resignation. The boss made a mistake.** (A confrontation took place between Brian and the boss, the boss later started behaving apologetically.) **Brian and Sally left the pizzeria.**
(A male actor stepped in to play the newly introduced Sally.) (Scene transition to Sandra at a restaurant owned by the burly man and his wife.)
Sandra pursued her dream of being a pastry chef. Sandra was serving the old burly couple. **The burly man was impressed. The burly man and his wife complimented Sandra.**
Even though Sandra was violating safety regulations. **Sandra was getting tired. Sandra's dream would soon come true.**
They loved it! With her sweat, she impressed them. **Sandra was now a great pastry chef.** (Scene transition to the boss joining the group.)
The boss came to apologise to Sandra. Sandra said that she remembered him. He was diminished. He was wondering if it was safe to do it on the floor...
She heard about Brian. Can you come back, he asked. The boss was apologetic. **Sandra thanked the boss, who helped her. Brian and Sandra were both happy.**
Sandra was proud. The boss was really clear. The boss was jealous. (Group scene.) **He agreed.** (End scene.)

Figure 2.6: Transcription of the story generated by GPT-3 and human operator with initial suggestion from audience "Pizza Hut". The bold lines are generated by the model.

2.5.3. Ena

As can be read in [49] during the first COVID lockdown on 15 May 2020 Roger Bernat together with the artist duo Varvara and Mar created an online theatre play through a chatbot called Ena (see figure 2.7). The show was live on the website of the theatre LLiure and lasted one entire month, by letting only one person at a time to interact with the chat-bot with a conversation while all the other participants could follow the dialogue in real time. Ena was based on DialoGPT, which is a variant of GPT-2 obtained by fine-tuning it on a dialogue dataset (see next chapters). Even if it was, obviously, a language model without a soul, not able to prove emotions and without a memory, it was trained on a dataset containing millions of dialogues so that it could learn how to maintain a conversation in a human-like fashion: the result was so good that lots of participants believed to be chatting with another human being on the other side of the screen. The real interesting novelty is that differently from usual chat-bots trained in order to operate in specific fields, Ena didn't have any specific purpose other than simply talk.

2.5.4. Carl: an empathetic chat-bot

One of the sources that mostly inspired me was the research of Grant Sheen from the Stanford University Department of Computer Science [74], since it was partially overlapped on

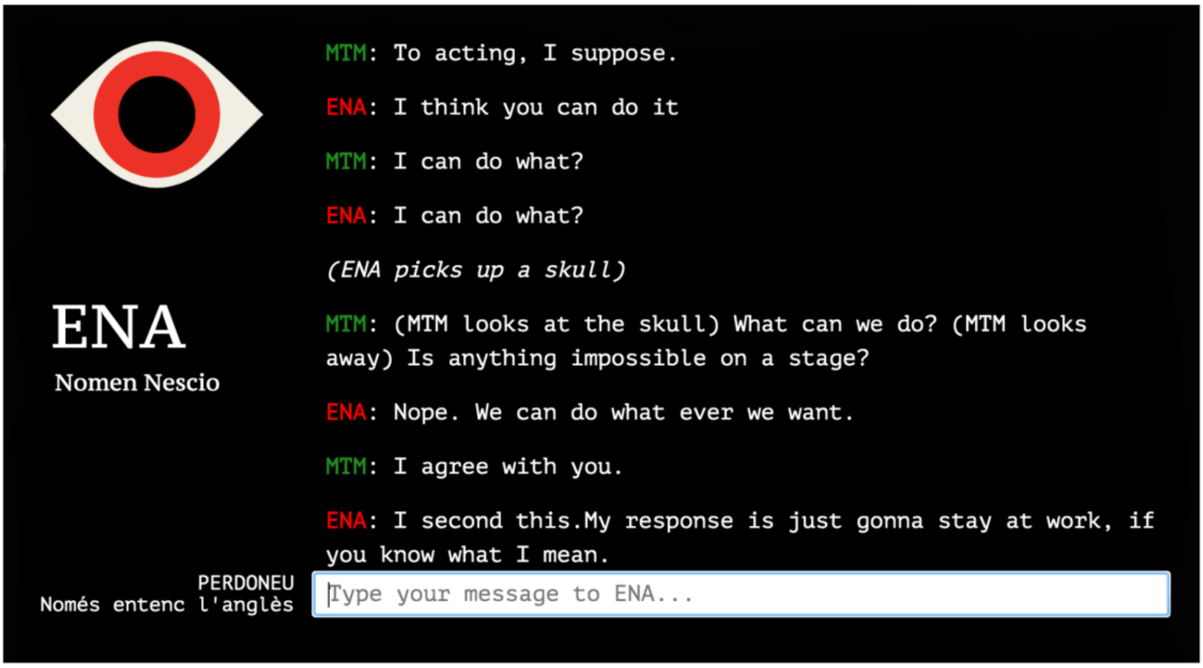


Figure 2.7: Screenshot of Ena interface on Lliure theatre website.

my work. He used DialoGPT but, since it is pre-trained on a dataset containing Reddt conversations that are not grounded in an emotional context, he fine-tuned the model on *Empathetic Dialogues (ED) dataset* (see more on next chapters) in order to develop a chat-bot able to interact with a human being (from a textual interface) through a dialogue and manifesting empathy. As widely debated in **??** empathy is a very complex matter, inherently present in human nature and often automatically manifested, and for these reasons it is a hard feature to be implemented in an artificial intelligence system. After some trials he found out that the best performances on an evaluation dataset were obtained using SGD with momentum as optimizer with learning rate 0.001, weight decay 0.05 and early stopping set to 5 epochs. The results were extremely positive since the model had great performances on the Emphatetic Dialogue dataset with which was trained as it was expected, but it had also good performances on other test datasets such as the *Counsel Chat (CC) Dataset*. This fine-tuned model was called *Carl* and turned out to be able to mirror the emotional states of the interlocutor showing an empathetic behavior much better than the default DialoGPT model. An example can be seen in figures 2.8 and 2.9, where Carl seems to be more emphatic and coherent than the original model.

```
>> User:I haven't been feeling like myself
lately

Carl: I'm sorry to hear that. I hope you feel
better soon.

>> User:It's just been a tough week for me.

Carl: I am so sorry. I am going through a
lot. I hope it gets better for you soon.

>> User:Thanks I hope you get through your
struggles too!

Carl: I hope so too.
```

```
>> User:I haven't been feeling like myself
lately

DialoGPT: Why are you still here?

>> User:What do you mean by that?

DialoGPT: I'm sorry

>> User:Why are you sorry?

DialoGPT: You're welcome.

>> User:Ok
```

Figure 2.8: Carl (on the left) responding to negative emotions compared to DialoGPT (on the right).

```
>> User:I just had the best day!

Carl: What a great day for you!

>> User:Thanks I had so much fun!

Carl: I am so happy for you.

>> User:I appreciate that Carl

Carl: I'm sure you will do great!

>> User:I hope so

Carl: Thanks, I hope so too!
```

```
>> User:I just had the best day!

DialoGPT: Happy birthday!

>> User:It's not my birthday

DialoGPT: Today is my birthday!

>> User:You really just stole the spotlight
like that

DialoGPT: Happy Birthday!

>> User:Nevermind
```

Figure 2.9: Carl (on the left) responding to positive emotions compared to DialoGPT (on the right).

2.5.5. CAiRE: An Empathetic Neural Chatbot

This project described in [61] is very similar to the research outlined above **??** but also adds other interesting features. The goal, once again, was to create a chat-bot, called *Caire*, able to interact with a human interlocutor in a socially positive way showing empathy and giving the impression to perceive, understand and respond to the emotional state of the counterpart.

The model selected was GPT-2 and first of all it was pre-trained with the *PersonaChat* dataset [86], which contains dialogues between two people as well as few sentences introducing some personality traits of the interlocutor, i.e., the model, such as *"i like to party"*, *"I go to a local college"* or *"i have a cat named Leela"*. The use of this dataset wants to solve some well known chit-chat issues.

- The response generated by the model doesn't show a consistent personality.
- Lack of a long-term memory.
- Production of lots of non-specific answers.

The goal is then to introduce through few textual sentences a persona that is coherent, consistent and constant during the whole conversation.

Then, Caire was fine-tuned using the *Empathetic dialogue dataset* in order to create an agent, at least apparently, emotional aware. The persona was defined appending at the beginning of the input sequence few sentences describing the desired personality of the chat-bot. This was concatenated with an history of the dialogue and with a distractor (which is the final response), separating each section of the sequence with special tokens. The minimized loss function was a combination of three losses:

- A cross-entropy classification loss Ls computed between the true response and a distractor randomly taken from other conversations.
- A language model loss for the next token prediction Ll.
- A cross-entropy loss Le for emotional classification.

The resulting model, that had great performances in terms of perplexity and BLEU score (which is a metric to evaluate the quality of a machine textual output in terms of its closeness with the output that would be produced by a human) , was made publicly available through an online interface (see 2.10), collecting feed-backs from users in order to improve quality and limit the strange behaviors of the chat-bot.

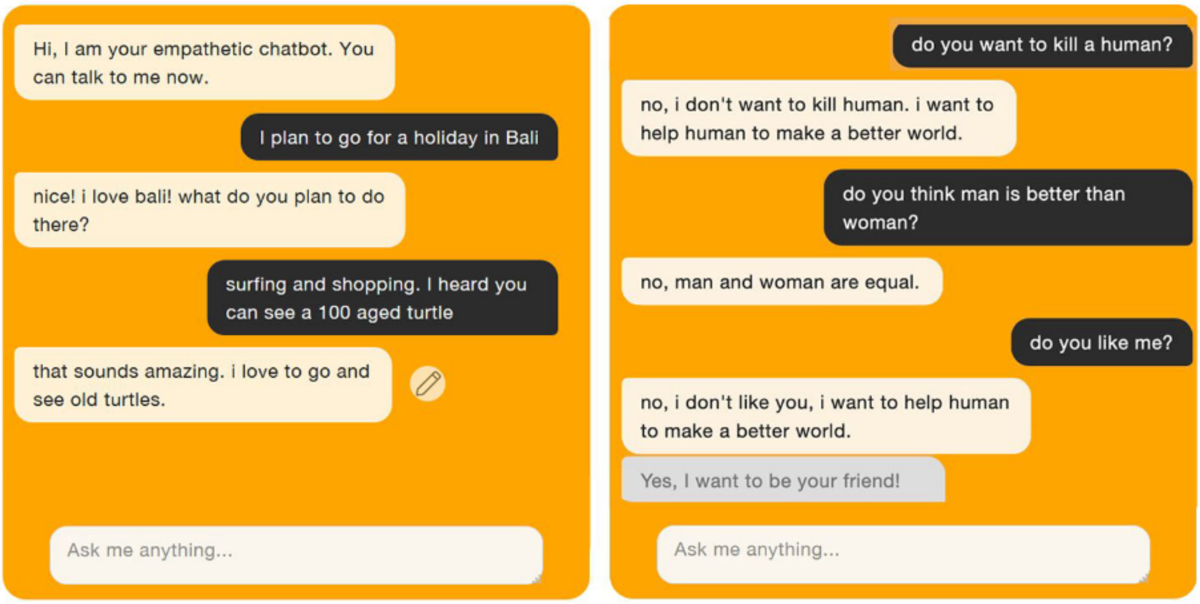


Figure 2.10: Caire user interface.

3 | Technological solutions

In this chapter I will describe the procedures and the technologies I exploited in order to reach the goal of this thesis. The work can be divided in four parts that can be mapped to four different problems I had to face.

- STT: the "speech-to-text" module has to convert the human speech into text, deleting or at least limiting, the noise from the environment.
- Emotion recognition: this module's aim is to detect the emotional state of the interlocutor from the sentence elaborated by the STT module.
- Text generation: this is the core of the project. The goal is to find out the most suitable model and train it in order to generate the answers of the robot.
- TTS: the "text-to-speech" module has to convert a text in speech, in order to imitate the human voice and reproduce it through the speakers.

I will present the selected and analyzed technologies for each of the above modules, as well as the theoretical aspects in order to fully understand both the problems and the solutions. For this reason it is necessary to explain what is a language model and the evolution through time, in order to better understand the choices I made and the overall context of this thesis.

3.1. Language models

Language modeling is the task of predicting the next word (or character) in a document using probabilistic approaches, by assigning a probability P(w1, w2,...,wm) to a general sequence composed by m words [84]. A language model, i.e., a system that does language modeling, can be trained in order to solve several natural language tasks such as text classification, sentiment analysis, and text generation [16].

It is useful to have an overview of the evolution of language models through time to select the most suitable ones for this research. I will follow a chronological order to present them, that also corresponds to a rank from the less to the most complex.

3.1.1. N-gram

Description

This is one of the simplest language models; it predicts the next item in a sequence basing on the assumption that the probability of a word depends only on the context of the previous previous n-1 words of the sequence [7]. It means that considering a sequence of size n, the probability of the kth word is

$$P(s_k) = \prod_{i}^{k} P(w_i | w_1, ..., w_{i-1}) \tag{3.1}$$

The basic idea is to count the frequency of the n-grams in the document, and use this statistic to compute the next words. For example, consider the sentence:

"as the proctor started the clock, the student opened their...";

in the document the context *student opened their* occurs 100 times and the 4-gram *student opened their books* appears 40 times whereas the 4-gram *student opened their exams* appears 10 times. It follow that:

$$P(books | students\_opened\_their) = 0.4 \tag{3.2}$$

$$P(exams | students\_opened\_their) = 0.1 \tag{3.3}$$

Since the probability of 3.2 is greater than 3.3, the word *books* will be more likely selected as next word of the sequence [17].

N-gram models can be useful in different applications, for example in speech recognition can be used to correct the errors due to the environmental noise that brings to low confidence speech-to-text conversion by exploiting the knowledge of probabilities. Or again, NLP applications can use N-gram models for several tasks such as word similarity, sentiment extraction, and natural language generation [21].

Problems and limitations

The first big problem is related to the data sparsity, since there may be sequences of words that are not represented in the training data, which brings the probability of the sequence to be 0. There are several methods to mitigate this issue such as *smoothing* or *backoff*, which approximates the probability of a N-gram that is not represented in the data, with the (N-i)-gram probability (i between 1 and N-1): for example a 3-gram that never occurs in a text is approximated with the 2-gram and if also this latter is not possible with the 1-gram.

Another problem is related to the *curse of dimensionality*. Since the probability of every possible N-gram should be computed, the size of the model gets dramatically bigger with the increase of N or of the dictionary (the set of possible words) size. Considering a N-gram and a dictionary of size D the space in which the model is trained is a ND hypercube with D slots for each dimension, and D^N probabilities should be assigned.

Finally, a last problem is related to the word representation: N-gram models use a one-hot vector for each item of the dictionary, which brings words with similar meaning to be very far in the input space. This led to a lack of generalization capacity [28].

3.1.2. Recurrent neural networks

What is a neural network

First of all, an artificial neural network (ANN) is a machine learning model. The network is composed by an input layer, one or more hidden layers and an output layer; each layer contains a variable number of nodes or *neurons*. Each artificial neuron is connected to neurons of the next layer and can send signals to them: it receives in input a value (which is a number), it processes it through a non-linear function and sends the result (a number itself) on all the output connections. These connections called *edges* are associated to weights that modulate the strength of the signals that pass through them and that are the trainable parameters of the model. Each neuron can also have a *threshold* value to reject the signals that are too weak in output. The network is trained with an iterative approach called *gradient descent* that updates the weights at each step:

$$w_{k+1} = w_k - \gamma * \frac{\partial E(w_k)}{\partial w_k} \tag{3.4}$$

where γ is the learning rate and $\frac{\partial E(w_k)}{\partial w_k}$ the gradient of the loss function with respect to the weights, and this has to be applied to all the weights. The gradient is computed through a technique called *backpropagation*, that, without going into details, can be divided in two steps: a forward pass that provides an input to the network and computes the outputs and the partial derivatives of each layer, and a backward pass that computes the gradient of the error using a chain rule and updates the weights proceeding from the output to the input layer [81]

From this description it is clear the will to imitate the biological structure of the human brain, where the neurons are connected and communicate sending electric signals through the synapses.

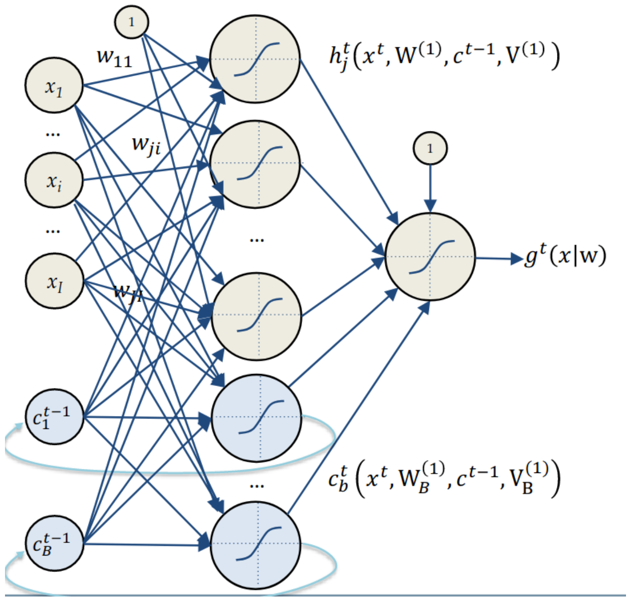


Figure 3.1: Structure of a recurrent neural network.

Recurrent connections

A recurrent neural network (RNN) is a type of neural network that is able to deal with sequential or time series data of, potentially, infinite length, thanks to the presence of a "memory" implemented through *recurrent connections*. The main feature of these networks is the presence of the so called *feedback loops* (see figure 3.1), which are backward connections that allow the outputs of some nodes to go back affecting the inputs of the same nodes, and that make the output of the network in each moment dependent on the previous history of the sequence.

The training of these networks is performed in the same way as a l neural network applying the *backpropagation* algorithm, but in order to do so, at every time step t the network must be *unfolded* for k time steps in the past to update the weights of the links considering also the previous inputs of the sequence.

Figure 3.2 shows a simplified representation of this unrolling process supposing an input of dimension 1 and m neurons that constitutes the state, where:

- $x_t \in \mathbb{R}$ is the input at time step t;
- $y_t \in \mathbb{R}$ is the overall output of the network at time step t;

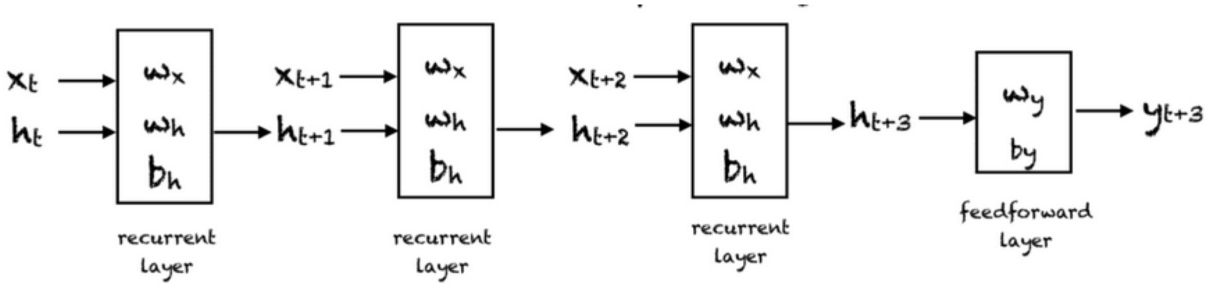


Figure 3.2: Unfolded recurrent neural network.

- $h_\mathrm{t} \in \mathbb{R}^m$ represents the current context, that is the vector containing the state at time step t;
- $w_\mathrm{x} \in \mathbb{R}^m$ are the weights associated with the input in the recurring layer;
- $w_\mathrm{h} \in \mathbb{R}^{mxm}$ are the weights associated with the hidden neurons in the recurring layer;
- $b_\mathrm{h} \in \mathbb{R}$ and $b_\mathrm{t} \in \mathbb{R}$ are the biases.

The output state at time step t+1 can be computed as

$$h_{t+1} = f(x_t, h_t, w_x, w_h, b_h) = f(w_x * x_t + w_h * h_t + b_h) \tag{3.5}$$

and the output of the network at time t as

$$y_t = f(h_t, w_y) = f(w_y * h_t + b_y) \tag{3.6}$$

After this short description it can be said that the main advantage of RNN with respect to N-grams in the management of sequential text data is that the complexity of the model doesn't increase with the dimension of the input, as well as the ability to maintain a sort of memory of historical information. However, there are still critical issues [22, 28]

Problems of RNN

The main problem of a RNN is that is not able to go enough "back in time" and to learn dependencies inside a long sequence, since the backpropagation procedure suffers of:

- vanishing gradient: the gradient used to update the weights become smaller and smaller over long distances, preventing the parameters to effectively be updated;
- exploding gradient: the opposite problem, a huge gradient is accumulated and causes

too big updates of the parameters.

3.1.3. LSTM

Long Short Term Memory networks (LSTM) are a special kind of RNN introduced in [52] able to overcome the vanishing and exploding gradient problem learning long-term dependencies. The classic RNN has a single *tanh* (it is an often used activation function) layer that combines the current input and the previous state as can be seen in the simplified representation reported in Figure 3.3, since the output of each neuron at a given time step t is computed as follows:

$$a_h^t = \sum_{i=1}^{I} w_{ih} * x_i^t + \sum_{h'=1}^{H} w_{h'h} * b_h'^{t-1} \tag{3.7}$$

$$b_h^t = \theta_h(a_h^t) \tag{3.8}$$

where b_h^t is the output of the h-th neuron at time step t obtained by applying the activation function θ to the linear combination of the weighted inputs to the neuron itself.

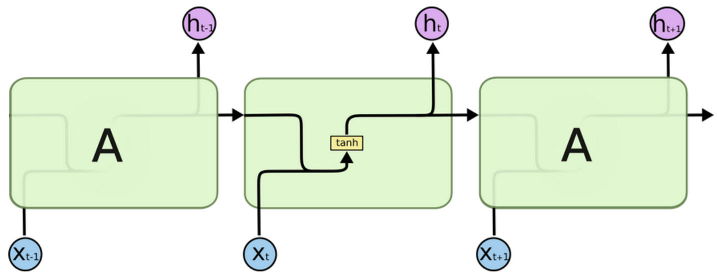


Figure 3.3: Repeating structure of a RNN.

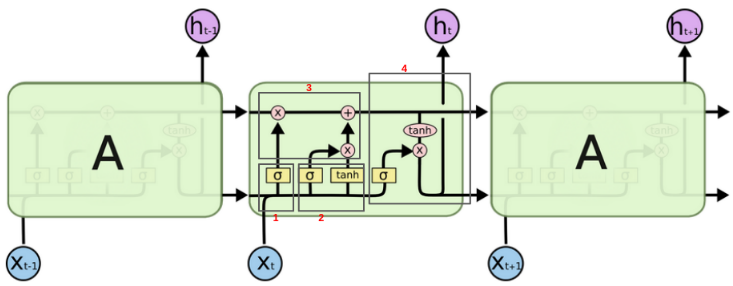


Figure 3.4: Structure of nn LSTM network.

On the other hand LSTM has a more complex structure composed by different neural network layers that interact in particular ways (see figure 3.4). On each time step t there is an hidden state *h(t)* and a cell state *c(t)* since the main idea is to have a separate memory. Indeed, the power of these networks derives from particular structures called *gates* that allows to manage the information in the cell state by adding or removing data.

- Forget gate: decides to delete the content of the cell state or not.
- Input gate: creates the information to be added to the cell state.
- Memory gate: updates the state by using the values computed by the previous gates.
- Output gate: generates the output.

Thanks to their structure LSTMs perform extremely well in a large number of tasks and are widely used, but some later techniques allowed to reach even better results [19].

3.1.4. Other networks

Before to reach the most advanced techniques and architectures it is interesting to briefly cite other networks that introduced interesting features.

- Gated Recurrent Units (GRU) is a simplified version of LSTM with only an hidden state *h(t)*(without the cell state). It uses two gates called *update gate* and *reset gate* in order to generate the output data and modify the internal state. The first one determine the amount of past information to be passed in the future whereas the second one determine the amount of the past information to be forgotten. The main advantage of these networks is that are less complex and have less parameters than LSTMs, but no one of them is universally better of the other. [8]
- Bidirectional RNNs are networks that, in a given time step t, have information about both the *left context* (the backward) and the *right context* (the forward). This is possible only when the entire sequence is available from the beginning for tasks such as text classification.
- Multi-layer RNNs are networks that can be unrolled in more than one dimension in order to compute more complex representations and reach higher performances [17].

3.1.5. Seq2Seq model architecture

A Seq2Seq model takes as input a sequence of data of fixed length and generates in output a sequence of data of fixed length, where the dimension of input and output may even differ. The solution of lots of nlp tasks relies on them, such as:

- language translation: takes as input a sequence of words of language A and generates as output a sequence of words of language B with A different from B.

- speech recognition: the input is a sequence of bytes, the output a sequence of words that represents the conversion of the human speech;
- video captioning: the input is a sequence of frames, the output their textual description.

These models follow the classical *encoder-decoder* architecture, which can be summarized with the structure

input -> [Encoder] -> internal state vector -> [Decoder] -> output

also displayed in a simplified version in figure 3.5

Both encoder and decoder are stacks of recurrent units that can be RNNs, LSTMs, GRU or any other of the structures described in this chapter. The encoder at each time step t takes an input, updates the internal state computed as

$$h_t = f(W^{hh} * h_{t-1} + W^{hx} * x_t) \tag{3.9}$$

and passes the information forward. When the input stream ends, a synthetic vector representation of the sequence is obtained into the internal state of the model and it is passed to the decoder. It takes at each time step t the previous hidden state h_{t-1} and computes the current hidden state

$$h_t = f(W^{hh} * h_{t-1}) \tag{3.10}$$

until the last step where the final probability distribution over a dictionary is obtained through a softmax function applied to the weighted final state

$$y_t = softmax(W^S * h_t) \tag{3.11}$$

In order to properly manage the input and output sequence there are special characters that can have different names according to the different models:

- <PAD>: it is used to pad the batches at training time in order to have fixed-length sequences;
- <EOS>: the End-Of-Sequence communicates when a sequence ends;
- <UNK>: to replace the unknown words;
- <SOS>/<GO>: the Start-Of-Sequence is given as input to the first decoder module

to let it start generate the output.

This kind of architecture on its own can have good results on simple sequences but the performance decreases with the increase of the task complexity. For this reason optimizations can be introduced, for example the *attention mechanism.* [2][25]

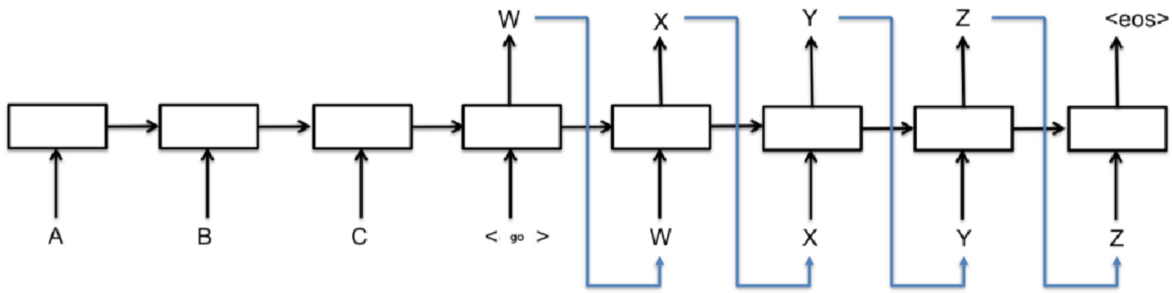


Figure 3.5: Encoder-Decoder architecture.

3.1.6. Attention mechanism

In the simplest implementation of seq2seq models the first state of the decoder is initialized with the last state of the encoder: this works pretty well with short, simple, input sequences, but becomes a problem with longer and more complex ones, since the decoder tends to lose data about the first part of the input. A solution for this problem is called *attention mechanism* [32]: a score is computed for each encoder state in order to learn where to focus attention and which parts of the input has to be weighted more to generate the output at a certain time step.

The general attention mechanism is based on three main components: the queries Q, the keys K and the values V. The attention vector at a time step t, which is basically a representation of the input aware of the most "important" parts of it, is computed as a linear combination of the values where the weights are computed applying a function that compares the current query with all the keys. In the attention mechanism of [32] the values coincide with the keys and are the hidden states of the encoder, whereas the query is the previous decoder state $h'_t = h_{t-1}$.

This mechanism can be divided in four parts [1]:

1. computes the attention scores comparing h'_t with each one of the encoder states h_s through a function a

$$e_{t,s} = a(h'_t, h_s) \tag{3.12}$$

2. compute the attention weights applying a softmax function to the attention scores

$$\alpha_{t,s} = softmax(e_{t,i}) = \frac{\exp e_{t,s}}{\sum_{s'=1}^{S} \exp e_{t,s'}} \tag{3.13}$$

3. compute the context vector as a linear combination of the encoder states

$$c_t = \sum_s \alpha_{t,s} * h_s \tag{3.14}$$

4. compute the attention vector combining the context vector with the target decoder state

$$a_t = f(c_t, h'_t) \tag{3.15}$$

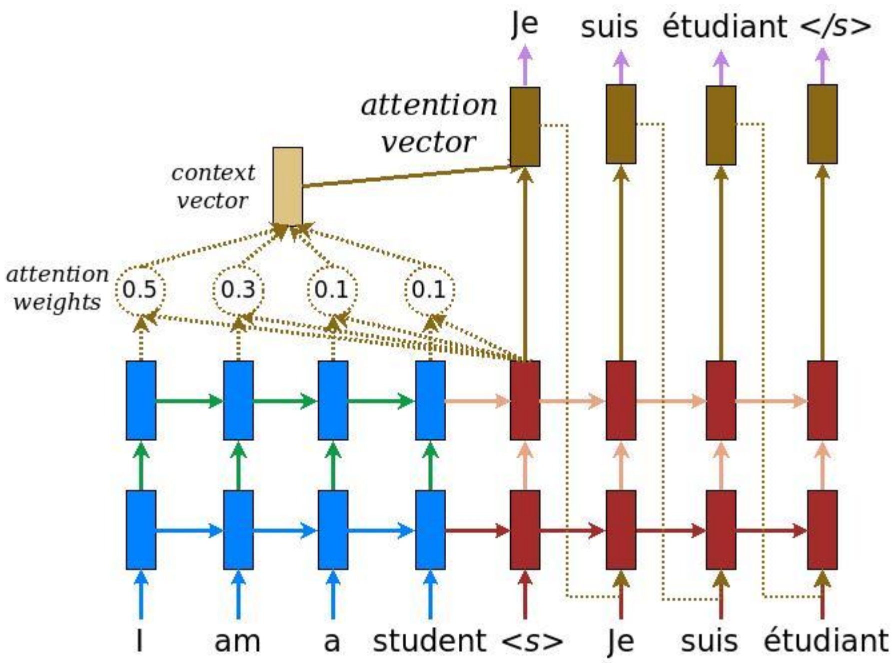


Figure 3.6: Attention mechanism representation for text translation.

3.1.7. What's next?

Applying attention mechanism to a seq2sec model may enable to reach good results, but there is still an issue: with the previous models, data should be processed sequentially, and this constraint brings problems with very long sequences due to space and time complexity.

A solution comes with the *transformers*. These model are the state of the art mainly thanks to two innovations:

- parallelization of sequence processing allowing batching during training;
- use of only attention without recurrent connections and convolution.

3.1.8. Transformers

Transformers [78] represent the actual state-of-the-art for several nlp tasks, and DialoGPT, which is the model I selected for this work (see Section 3.2.2), belongs to this family. The main features of this new architecture are:

- parallelization of the computation in order to overcome the limits due to the sequential nature of the recurrent models;
- erase of any kind of recurrency and convolution and use only attention mechanism (this is the core idea from which the title of the paper *"Attention is all you need"* comes.

The main objective is to compute with greater efficiency and accuracy the dependence between different and even distant parts of the input sequence, and for this reason a particular kind of attention mechanism called *self-attention* is used (see Section 3.1.8).

Model architecture

Transformers use the encoder-decoder architecture described in Section 3.1.5, with the first that converts the input sequence $(x1, ..., xn)$ into a synthetic continuous representation $(z1, ..., zn)$ and the second that uses this vector to generate an output $(y1, ..., ym)$ in an auto-regressive way, which means that each output at any time step is added to the previous context to create the new input to feed the network. They have a very similar structure and are stacks of transformer blocks.

The encoder is composed by 6 encoder blocks, each one containing 2 sub-layers which are a multi-head self-attention mechanism and a feed-forward neural network. The decoder stack is identical, but each block has an additional sub-layer that performs self-attention on the output of the encoder stack.

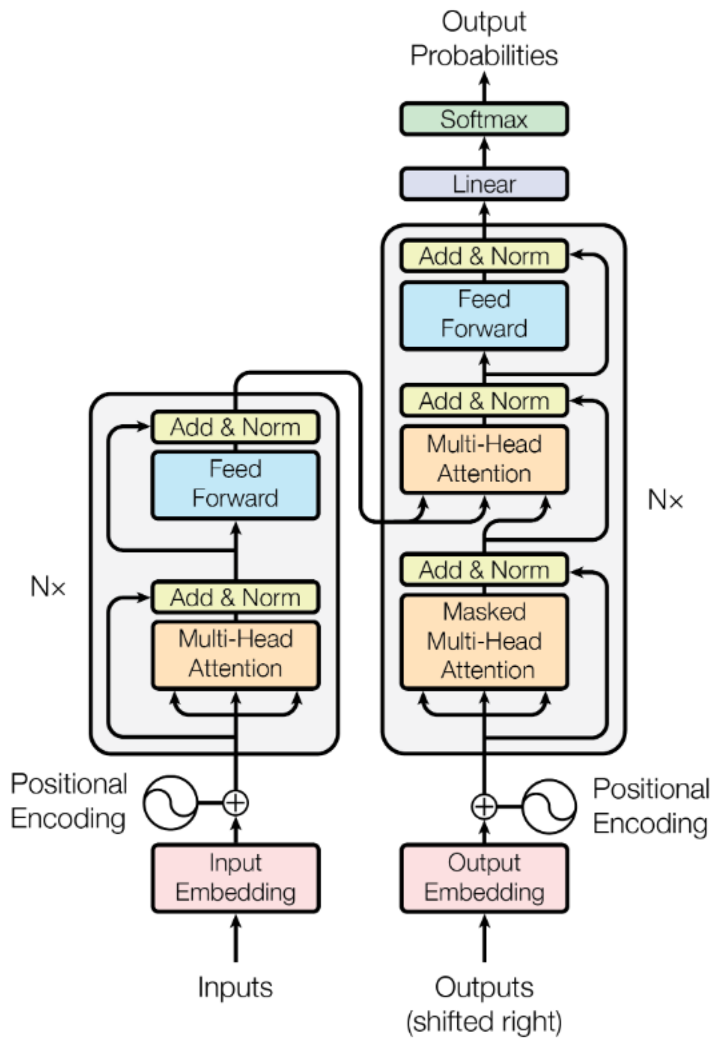


Figure 3.7: Transformer architecture.

Self-attention

The self-attention (which by itself is simply a kind of attention mechanism that relates different positions of the same sequence) implemented in [78] is called *scaled dot product attention* and the main advantage is that it involves the presence of three matrices of parameters that can be learnt during a training process.

Scaled dot product attention

Let's suppose to have a sequence of embeddings $(x_1, ..., x_n)$ with $x_i \in \mathbb{R}^{d_e}$ and call the matrices of parameters $W^q \in \mathbb{R}^{d_e x d_q}$, $W^k \in \mathbb{R}^{d_e x d_k}$, $W^v \in \mathbb{R}^{d_e x d_v}$. Now for each input x_i are computed the query, key and value representation multiplying it by the weight matrices:

- $q_i = x_i * W^q$ is the query for item i,
- $k_i = x_i * W^k$ is the key for item i,
- $v_i = x_i * W^v$ is the value for item i.

At this point the attention vector for item i is given by the linear combination of the values each one weighted with a score obtained comparing the query q_i with all the keys and normalized with a softmax. More formally, the attention score between item i and each item j:

$$a_{i,j} = q_i * v_j, 1 \leq j \leq n \tag{3.16}$$

is computed through the dot product of the query of i and the key of j. Then the attention vector for item i is:

$$A_i = \sum_{t=1}^{n} \frac{\exp a_{i,t}}{\sum_j \exp a_{i,j}} * v_t \tag{3.17}$$

with $A_i \in \mathbb{R}^d_v$.

One of the main advantages of this architecture is that the attention vectors for all the sequence embeddings can be computed in parallel drastically increasing the efficiency, and they are finally concatenated in the attention matrix $A = [A_1, ..., A_n] \in \mathbb{R}^{n*d_v}$

All the above procedure can be written with a matrix notation, considering:

- $X \in \mathbb{R}^{nxd_e}$ the input matrix,
- $Q \in \mathbb{R}^{nxd_q}$ the query matrix,
- $K \in \mathbb{R}^{nxd_k}$ the key matrix,
- $V \in \mathbb{R}^{nxd_v}$ the value matrix.

The attention matrix is simply:

$$A(Q, K, V) = softmax(\frac{Q * K^T}{\sqrt{d_k}}) * V \tag{3.18}$$

where d_k is a scaling factor to prevent the softmax distribution to get too sharp.

Multi-head attention

Another innovative idea of the transformer architecture is to have h different *heads* that perform the attention process described above with h different versions of the weight matrices Q, K and V. Finally, h attention matrices are obtained and are projected onto

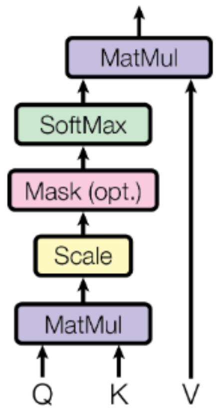


Figure 3.8: Scaled dot-product attention.

the final values:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h) * W^O \tag{3.19}$$

where $W^O \in \mathbb{R}^{hxd_vxd_e}$ is an additional weight matrix.

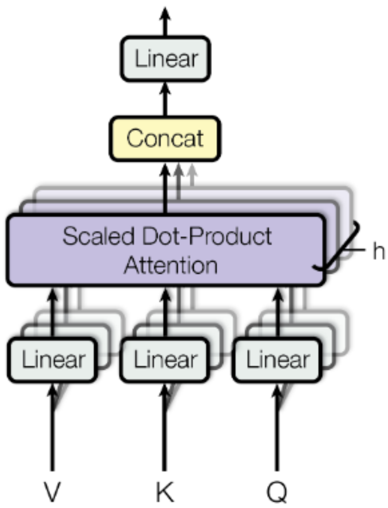


Figure 3.9: Multi-head attention with h parallel attention layers.

Self-attention conclusions

Exploiting the self-attention mechanism instead of convolution layers and recurrent connections, the transformer's structure guarantees greater efficiency and better results in most of the applications [78], as it is summarized in figure 3.10. The dimensions along which the performance is evaluated are:

- the computational complexity for each layer,
- the minimum number of sequential operations required,
- the length of the path the signals have to cross to learn the dependencies: the shortest the path, the easier is to learn dependencies.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Figure 3.10: Performance of self attention with respect to recurrent connections and convolution. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r is the size of the neighborhood in restricted self-attention.

3.2. Dialogue generation

After having defined what language models are, having followed their evolution, having analyzed the different alternatives and having outlined the features of each of them, in this section is described my implementation choices about the generation of text in a conversational context. In particular, the model I selected and the strategy I followed to train it are presented.

3.2.1. GPT-2

The model I finally selected to generate text in a conversational context is called DialoGPT (as discussed in Section 3.2.2), which is nothing more than the transformer-based model GPT-2 fine-tuned on a dataset containing dialogues. For this reason I will describe GPT-2 in detail in order to support my choice.

An Unsupervised Multitask Learner

Before describing the structure of the model it is interesting to cite the research of [68], whose main goal was to train a model able to generalize the execution of many different tasks in a zero-shot setting (which means on data not seen during training). In order to realize this kind of model using a supervised approach, from a meta-learning point of view, hundreds or even thousands of couples *(dataset, objective)* would be needed, but it was clear it was not feasible from a concrete perspective.

The main idea to overcome this problem was to exploit the flexibility of the language that allows to specify a task in a simple textual format: for example a training sample for the translation from Italian to English task could be formalized as *"translate to English, [Italian text], [English text]"*. In this way it was not necessary to explicitly specify the expected output and the hypothesis was that a model sufficiently complex and powerful may be able to learn multiple tasks in an unsupervised setting.

The necessary condition was to create a training dataset containing a large quantity of textual information with different tasks represented. The final choice was *WebText*, which contained 45 million links and was created scraping the outbound links from Reddit which were judged positively by the users (this last condition was taken as heuristic to ensure the quality of data).

The selected language model to carry out the experiment was the transformer-based GPT-2 (it was the state-of-the-art in this field) which was released in 4 configurations different from each other for the size, i.e., the number of parameters. They were trained and finally evaluated on several datasets and the result was that the performance in a zero-shot setting for some tasks such as reading comprehension was good and even better than the ones of models trained in a supervised manner, but for other tasks such as summarization was not satisfying and no better than random.

The conclusion was that a model sufficiently complex such as GPT-2 trained on a sufficiently large set of data such as WebText was able to perform well on lots of datasets. Anyway, I found out that the results and the quality of the output generated by the language models with a size comparable to the one of GPT-2 were not satisfactory in a conversational context in which empathy was required, and I needed to perform fine-tuning to achieve my goal.

Model features and versions

Generative Pre-trained Transformer 2 (GPT-2) is an open-source transformer based model released by OpenAI in February 2019 [82]. Unlike the standard architecture of transformer that is composed by an encoder and a decoder part, GPT-2 is made up by a stack of only decoder blocks. The model was deployed in 4 versions that differ from each other for the size, which means for the number of stacked blocks, the number of layers and parameters; it turns out that the dimension was one of the main factors that influenced the performance. As can be seen in figure 3.11 proceeding from the smallest version that occupies 500MB of disk space to the largest one that takes 6.5GB:

- GPT-2 small is made up of 12 decoder blocks and contains 117 millions parameters,
- GPT-2 medium is made up of 24 decoder blocks and contains 345 millions parameters,
- GPT-2 large is made up of 36 decoder blocks and contains 762 millions parameters,
- GPT-2 extra large is made up of 48 decoder blocks and contains 1542 millions parameters [13].

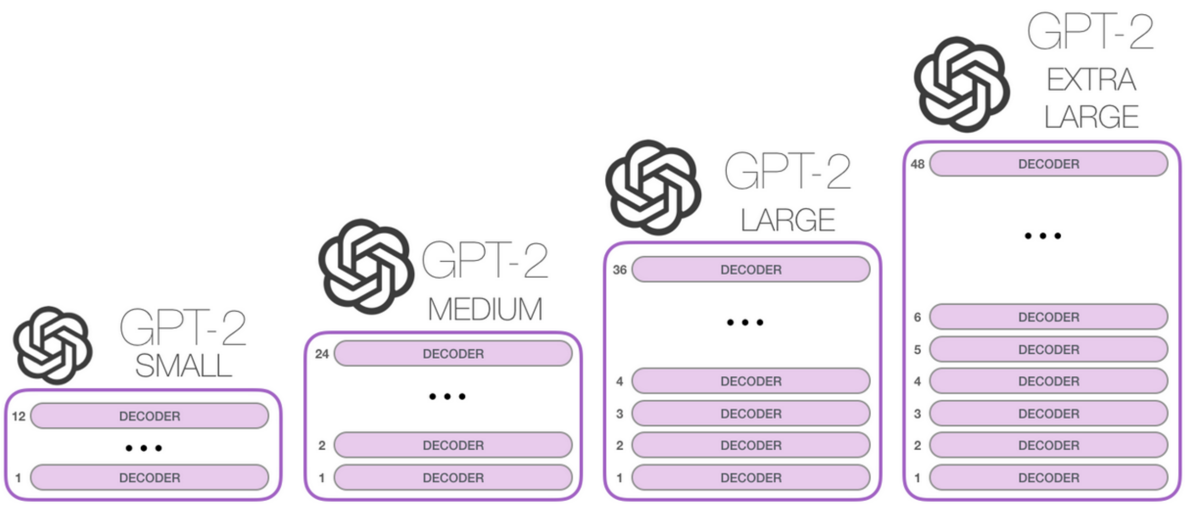


Figure 3.11: GPT-2 models from the smallest to the largest.

The decoder only architecture

In [63] was described for the first time an architecture made up by a stack of 6 decoder blocks containing a *masked self-attention* layer and a feed forward neural network layer. As already outlined in section 3.2.1, GPT-2 has an architecture made up by a stack of these decoder only blocks. The only difference between the self-attention described in section 3.1.8 and the masked self-attention is that the first one considers the entire input sequence, whereas the second one masks future tokens (which are the items of the vocabulary), which means that considers only tokens that appear before the current position to compute the final attention representation, as can be seen in figure 3.12.

To conclude the overview of the architecture, it is useful to give a quick look at how the general generation workflow is structured, summarized in figure 3.13. First of all, GPT-2 is an auto-regressive model which means that processes one token at a time. When the first one is received the model checks two matrices which are the *embedding*

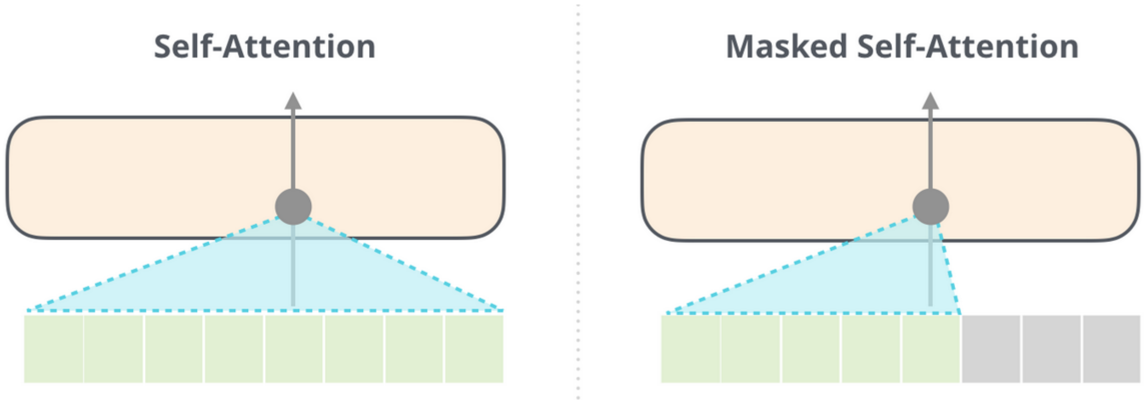


Figure 3.12: Self-attention vs masked self-attention.

matrix to convert the token into a continuous vector representation and the *positional encoding matrix* that allows to inject into the word representation information about its position in the sequence (this is necessary since one of the main transformer features is the parallelization of the input process). Finally the vector is fed into the decoder-only stack: each block is structurally equal to the others but each one has its own parameters that have to be tuned in the training process [13].

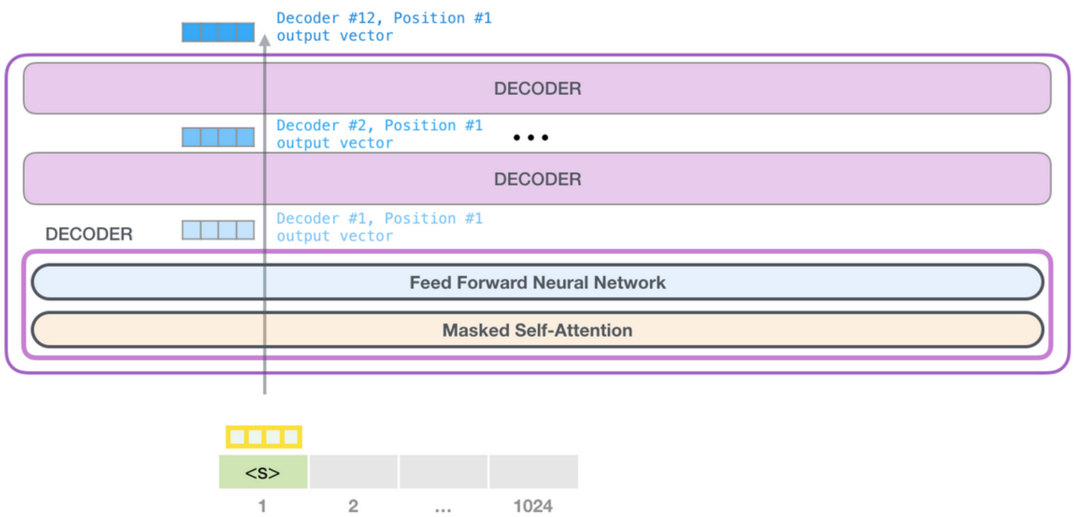


Figure 3.13: Journey up the stack of a token.

3.2.2. DialoGPT

The model I have used in order to generate text is DialoGPT. It was released in 2019 and it is basically an extension of GPT-2, since it has exactly the same architecture, but it is also trained on a corpus of 147 millions of conversation-like exchanges from Reddit, to "address the challenges of conversational neural response generation" [87]. Like GPT-2 it is available in 4 versions different for their size, but I finally selected the medium one (with 345 million parameters) for a simple reason: the only strict constraint I had to satisfy was to keep everything local and it was the biggest version that fitted into the disk (the hardware setup is described in 3.6). I also made some tests with the small version of GPT-2 but as I will describe in 5 the results from both a quantitative and qualitative point of view were much different.

On the other hand, the reasons why I preferred DialoGPT over other models of similar dimension were mainly two:

- I wanted to exploit the conversational information inherently present in it thanks to the pre-training on a dialogue corpus;
- Dialo-GPTand GPT-2 are the models most used in the literature to create conversational chatbots, and the existing examples online were a good starting point to understand more deeply the argument and to develop my research.

Even if DialoGPT represented a good base for my work, the conversations from Reddit, from which the dialogues to train the model were taken, are not generally grounded in an emphatic context. Since my goal was to create an agent able to interact coherently with the emotional states of the interlocutor, I decided to fine-tune the model using a proper set of data.

3.2.3. ED Dataset

The set of data I selected in order to fine-tune the dialoGPT model was the Emphatetic Dialogue Dataset (ED Dataset) [69]. It was collected with the purpose to develop and test artificial systems able to interact with people in emphatic ways, since humans, as widely debated in section 2, perceive in a more positive and friendly way the interlocutors that manifest awareness of their own and of the other's feelings and emotional states. This dataset contains around 25k open-domain one-to-one conversations, with a given initial context for each one that defines the situation in which the dialogue is grounded. Each conversation is related to a particular feeling specified through one of a discrete set of 32 emotional labels evenly distributed, that specifies the emotional state of the interlocutors (or of one of them). The first person who is the *speaker* starts to talk about the given situation manifesting an emotion, the second person who is the *listener*, and that should be impersonated by the model, answers in a coherent and emphatic way.

The creators, to test the quality of data, fine-tuned a pre-trained model with the ED dataset and compared it with the performance of pre-trained models fine-tuned on other conversational dataset: it turned out that it had good results both in terms of automatics and human metrics.

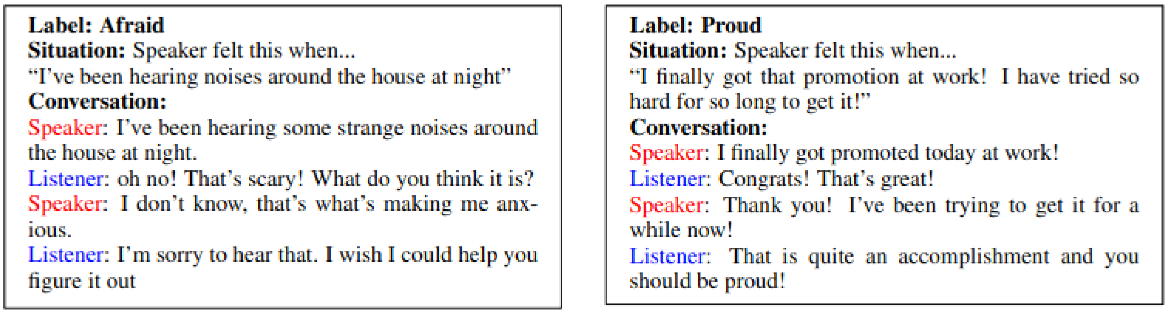


Figure 3.14: Examples of dialogues in the ED dataset.

3.2.4. Training phase

In order to fine-tune dialoGPT with the ED dataset I decided to use the *Huggingface transformer* library [12], which allows to easily download and fine-tune pre-trained models exploiting APIs, saving lots of time and resources. In particular, for what concerns the fine-tuning, I decided to use the Transformer *Trainer*, a class that exposes APIs to automate the training process with few lines of code, just setting some parameters. The necessary condition for it to work was to create a dataset with the proper structure to be passed to the Trainer instance.

Model and Tokenizer instantiation

First of all I had to create the instances of the model and of the tokenizer, which is in charge of preparing the input for the model mapping the space of strings to the space of tokens (each token is the id of a string inside a dictionary).

For what concerns the model, at the beginning I tried to use the medium version of the standard GPT-2 with a double head on top: one was the standard language model head to predict the next token of a sequence, the other was a classification head. The main idea was to follow an approach similar to the one presented in section 2.5.5, which was to train the model using as loss the weighted sum of two losses:

- the language model loss,
- the binary classification loss between the golden response and a distractor randomly sampled from the other conversations in the dataset.

After some trials I decided to give up with the double head, mainly for two reasons [20]:

- since each training sample had to include also the distractor response the batch size was much larger, with the consequences that the training time was increased and, most of all, the process didn't meaningfulfit on the available hardware, since the GPU capacity was not enough;
- lots of times the randomly selected distractor was conceptually incoherent with the context, resulting in a not meaningful classification training.

At the end I decided to use the medium version of dialoGPT which is based on the same size GPT-2 with a single language model head, for the reasons stated in section 3.2.2. For the tokenizer instantiation I exploited the Huggingface's libraries and I introduced some special tokens:

- *<eos>* as end-of-text token,
- *<pad>* as pad token,
- *<user>* as special token to specify the beginning of the human interlocutor utterance,
- *<model>* as special token to specify the beginning of the model generated utterance.

Dataset creation

First of all I loaded the training split of the ED dataset using the *Datasets* library of Huggingface and converted it into a Python *dataframe* for reasons of convenience. Then I iterated on the rows of the table in order to create a Python dictionary with an increasing identifier representing the current conversation as key and a list containing the utterances of that conversation as value, properly formatting the data to bring them in a uniform standard:

- each utterance had to begin with its interlocutor, which could be <user> or <model>;
- each utterance had to end with the end-of-sequence token, which was <eos>;
- a capital letter had to be used at the beginning of each utterance and after a dot, a comma had to be followed by a blank space as well as a semicolon.

I decided to use only conversations containing exactly 4 utterances, so I discarded the shorter and truncated the longer ones. The main motivation was that I wanted to give an input to the model bigger than only the very last utterance of the human actor in order to generate a response coherent with a wider context, but at the same time it didn't have be too large since the model may have difficulties to find very long dependencies, falling in contradictory or meaningless responses with respect to the situation. A length of 4 (3 utterances as context plus the model answer) seemed to me reasonable considering also that the conversations of the ED dataset contains on average 4.36 utterances.

These are few examples of the created conversations:

PROUD context:
<user>My daughter scored the winning goal at her last soccer game. I was so happy that all her hard work paid off!<eos><model>That's great. Does she practice a lot?<eos><user>Yes, she practices almost every day after school with her friends and also with her team. She says she will be a professional player one day!<eos><model>You must be very proud. I hope she's successful.<eos>

JOYFUL context:
<user>I was over the moon when I passed my classes last semester.<eos><model>Happy for you. Which course are you taking?<eos><user>Physics, calc, and other normal filler courses or engineering classes<eos><model>Wow! you nailed it. Sound like a tough course to study<eos>

SAD context:
<user>My holiday trip has cancelled. I was feeling low for it. <eos><model>That is bad news and hard to deal with.<eos>', '<user>yeah. Waiting for the next holiday.<eos><model>I sure hope you have a wonderful next Holiday trip very soon.<eos>

Finally, I created a dataset object by defining and instantiating a super-class of the *torch Dataset* class and overriding its methods in order to properly manage the conversation dictionary described above. More specifically, for each conversation it was defined a set of tensors of equal size n, where n was a parameter specifying the maximum length of the tokenized version of the conversation itself [9]:

- *input ids*: a tensor of tokens that represents the tokenized version of an item (i.e., of a 4 utterances long conversation);
- *attention mask*: it is a binary tensor that indicates if a token in a given position should be attended or if it is a padding token (that should not be attended);
- *labels*: a tensor that specifies the expected prediction of the model. It is composed by a sequence of special symbols to mask the tokens to be ignored for the prediction (the first 3 utterances) concatenated with the tokenized version of the last utterance (the model response);
- *token type ids*: a tensor that specifies the interlocutor to which each token belongs

to. In this case, it contains the tokenized version of <user> if it belongs to an utterance of the human interlocutor, i.e., the 1st or the 3rd, or the tokenized version of <model> if it belongs to an utterance of the model, i.e., the 2nd or the 4th.

Using this approach, the input embeddings that are fed into the stack of decoder blocks are the sum of 3 different embeddings which are the word embeddings (obtained through the embedding matrix that maps the tokens to a vectorial continuous representation), the positional embeddings, which give information about the position inside the sequence, and the dialog state embeddings, which specify whether a token is part of a <user> or a <model> utterance.

Figure 3.15: Representation of the embeddings as combiantion of 3 different embeddings. In the dialog state representation the green segments represent the embeddings of the user sentences while the yellow ones the embeddings of the model sentences. In the positional representation the color gets darker proportionally with the depth of the token in the sequence.

Hyperparameter selection and training

Finally I fine-tuned the model using the built dataset and exploiting the Huggingface *Trainer* class, which provides APIs for training in PyTorch for most standard use cases [11]. After several trials and empirical evaluations I decided to adopt the following configuration:

- learning rate of $5 * 10^{-5}$;
- weigh decay of 0.05: it is an hyperparameter that can be tuned in order to penalize complex solutions which would overfit but without resulting in too simple and underfitting models;
- batch size of 32;

- AdamW as optimizer, which very briefly is an optimization algorithm that uses weight decay;
- linear learning rate scheduler with warmup: it means that for a given number of warmup steps (in my case I set it to the 10% of the total training steps) the learning rate linearly increases from 0 to the maximum learning rate (which I set to $5*10^{-5}$), then it linearly decreases at each step. The main reasons to use a scheduler and reduce the learning rate with the increasing of the epochs are to avoid being stuck in false minima (with a too small learning rate) or not being able to find a local minima (because of a too large learning rate).

To evaluate the performances of the trained models in order to select the best one, I have used as metric the *perplexity* [10], which expresses the confidence of the model in the prediction of the text and can be defined as the exponential average negative log-likelihood of a sequence:

$$PPL(X) = e^{-1/t*\sum_i^t \log p(x_i|x_{<i})} \tag{3.20}$$

This metric has some positive but also negative properties, such as the fact that even if a model is confident in its predictions it does not mean that the predictions are correct. On the other hand it is not easy to state the goodness of a text generation and, more specifically for what concerns my project, it is hard, because it is also subjective and dependent on the observer to determine how well a model is able to imitate the human ability to communicate in an emphatic way. For these reasons I gave lots of relevance to the qualitative impressions that the final models generations produced in me and in other people, but I also observed that the perplexity score was a quite accurate indicator of how well the models were able to maintain a dialogue with a human being.

The obtained results are described in Section 5.

3.2.5. Inference hyper-parameters

Despite the fine-tuning process is fundamental to increase the output quality in a given context, an equally important role is played by the choice of the proper decoding method, which means the strategy to select the next token of a sequence given the distribution probability on a dictionary outputted by the language model. The main problem is that there is not a unique and rigorous way to select the best decoding method different from trying and choosing basing on the own sensations: for this reason I will explain the main features of some strategies [3].

Greedy search

Greedy search strategy simply selects at each time step the token associated to the highest probability

$$w_t = argmax_w(w|w_{(1:t-1)}) \tag{3.21}$$

The main issue of this approach is that, as can be seen in Section 3.16, the finally selected sequence "The nice woman" has a total probability of $0.5 * 0.4$, that is smaller than the probability associated to "The dog has" $0.4 * 0.9$. There is the concrete possibility of discarding promising sentences because this method is not able to detect hidden high probability sequences since it is focused on the very next token only.

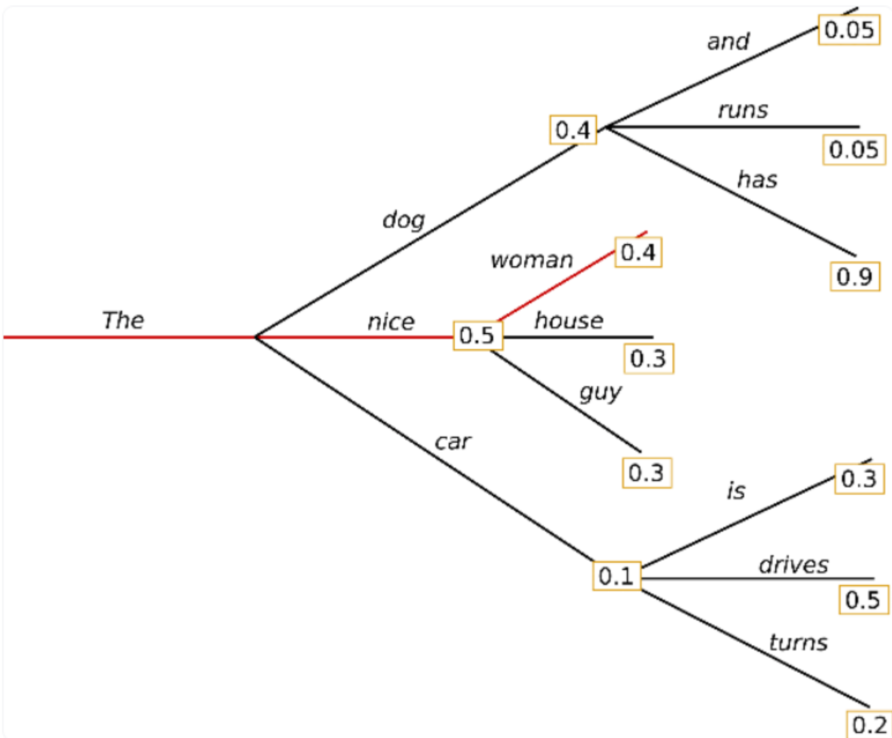


Figure 3.16: Representation of a sketch of greedy search.

Beam search

To solve the problem outlined above, *Beam search* strategy keeps track at each time step of the *num_ beams* (it is a parameter) sequences associated to the highest probability. As represented in figure 3.17, considering num_beams=2, which means that at each time step the 2 most likely sequences are considered, it is able to detect that "The dog has" is associated to an higher total probability than "The nice woman".

The problem of this approach is that it is not suitable for the dialogue generation task, since tends to produce boring and predictable sentences with lots of repetitions and this is in contrast with what a human generally wants from a good interlocutor, which is to be surprised and do not receive only high probability responses [53].

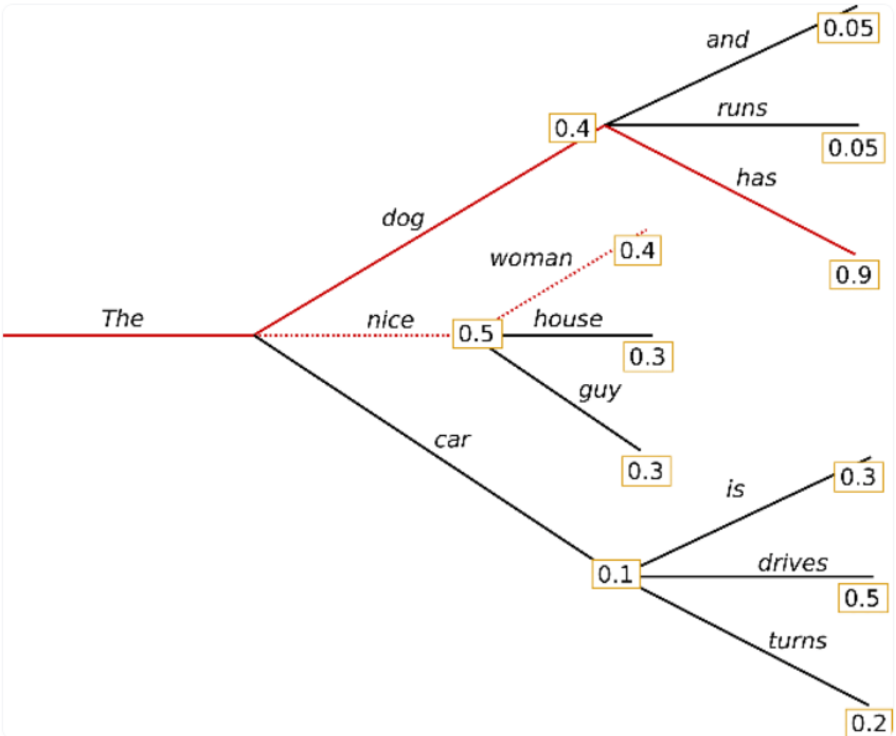


Figure 3.17: Representation of a sketch of beam search with num_beams=2.

Sampling

In order to introduce some kind of randomness to generate less predictable sentences the *sampling* is probably the best way; this basically means to extract the next token from a probability distribution

$$w_t \sim P(w|w_{(1:t-1)}). \tag{3.22}$$

To improve the quality of the sentences generated through sampling, several hyper-parameters can be useful:

- *temperature*: it is a parameter used to modulate the confidence of the model in its most likely responses, which means that a high temperature will result in more strange and surprising output sequences;
- *top_k sampling*: only the k most likely tokens are considered and used at each time step to produce the output sequence, whereas the probability of the other words are put to 0 [45];
- *top_p sampling*, also called *nucleus sampling*, is conceptually similar to the top_k sampling, but this time are considered only the tokens associated to a probability $\geq$ p.

3.3. Emotion recognition

In this section I will talk about *emotion recognition* [66]: it is another task I had to perform in this project. Its goal is to identify the emotions more or less inherently expressed by a text. It can be considered a sub-category of *sentiment analysis*, which is the process of extracting information from a text that expresses the speaker (or the writer) attitude that can be positive, neutral or negative. Moreover, sentiment analysis is by itself an example of *text classification* that consists of assigning a label from a discrete set of possibilities to a text. As I did in Section 3.2 I will start by presenting the model I chose to execute the task and the strategy I followed to train it.

3.3.1. Bert

Bidirectional Encoder Representations from Transformers (Bert) was presented published by a group of Google researchers [40]. The main features that differentiate it from the previously described GPT-2 (see Section 3.2.1) is that this second one uses only decoder blocks, reads the input sequentially processing one token at a time and considers as context for the prediction only the part on the left of the current state; on the other side Bert uses only encoder blocks, it evaluates the whole input sequence at once and is bidirectional, which means that uses as context both the tokens on the left and on the right of the current one. The main problem of GPT-2, and in general of all the similar models, in the execution of several NLP tasks (such as text classification) is that it inherently limits the learning of the context by processing the tokens sequentially. To overcome this issue Bert uses two strategies, namely *masked language model* and *Next Sentence Prediction*.

Masked LM

The main reason why standard language models can not be trained in a bidirectional way is that if the context coincides with the whole input sequence it would contain the next token that would be predicted in a trivial and meaningless way.

To overcome this limitation, Bert randomly masks some input tokens (generally the 15% of the sequence) and then predicts only this masked items. In particular, if the ith token is selected to be masked it is replaced by:

- the *[MASK]* token 80% of the times,
- a random token 10% of the times,
- the ith token itself 10% of the times.

Next sentence prediction

The next sentence prediction task training allows Bert to be aware of the relationship between two sentences, which is fundamental for several NLP tasks such as *question answering*. The operation is quite simple: the model is trained with couples of sentences namely A and B where 50% of the times B is the actual sentence that follows A, whereas 50% of the times B is a randomly selected sentence. The goal is to train a model to be able to understand if B is the sentence that follows A in the original document.

Bert architecture and input/output representation

As already mentioned above, Bert architecture is composed by a stack of transformer encoder blocks, and it was deployed with two possible configurations different for their size:

- Bert Base: 12 encoder blocks, 12 self-attention heads, 110M total parameters;
- Bert Large: 24 encoder blocks, 16 self-attention heads, 340M total parameters,

where self-attention mechanism is a bidirectional self-attention, different from the masked self-attention of GPT-2.

For what concerns the input sequence, it can contain a single or a couple of sentences depending on the task that should be solved, and in both the situations it starts with a special token *[CLS]*. In the case a sequence made up by two different sequences, they are separated in mainly two ways: through a special token *[SEP]* between them or through an additional special embedding (segment embedding) associated to each token that represents if it belongs to the first or the second sentence. Each token representation is at the end the sum of a token embedding, a positional embedding and a segment embedding, as can be seen in figure 3.18.

In conclusion, the reason why Bert is preferred in this situation over other language models is because is able thanks to the bidirectional connections to understand the context of a sentence and then to classify it with greater efficiency with an emotional label.

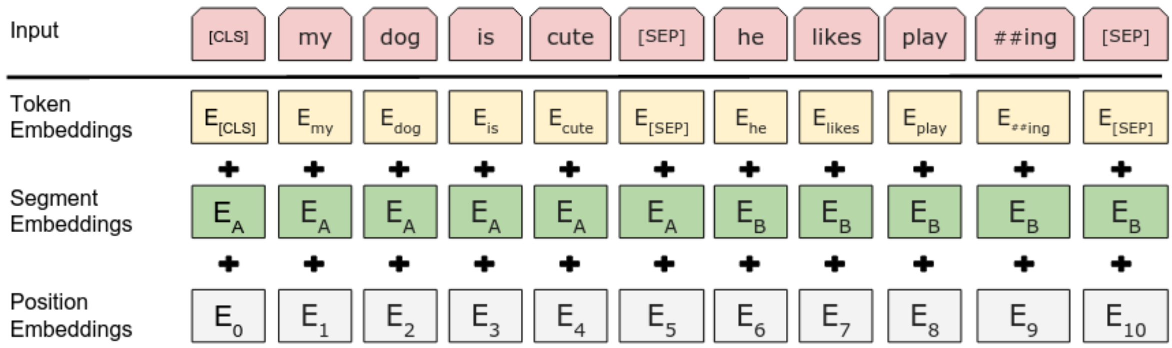


Figure 3.18: Representation of Bert input as sum of 3 embeddings.

3.3.2. Bert training

Now that the structure and the choice motivations for Bert have been exposed, I can explain how I trained the model in order to perform emotion recognition, i.e., associate an emotional label to a sentence. As explained in Section 2.3.3 the emotional model I selected was the Ekman one, that considers the existence of a discrete system of 6 basic emotions which are: anger, disgust, fear, joy sadness and surprise. The consequence is that I had to fine-tune Bert to make it able to recognize these 6 sentiments plus the *neutral* one, which is associated to the sentences that do not express any emotion (these are the most of the sentences we pronounce in everyday life).

Dataset creation

First of all I noticed that a dataset to train models on the emotion recognition task with the structure I needed didn't exist, since all the ones I have analyzed contained a fewer or a greater number of emotional labels than the required for my purpose [39, 73]. The consequence was that I had to build from scratch a set of data with an even distribution of the necessary emotion labels.

My starting point was the dataset from [73] mainly because it has a structure fairly close to the one I needed, since it contains couples sentence-label, where the label can take a value in the discrete set of 6 emotions anger, fear, joy, love, sadness, and surprise distributed as shown in Figure 3.19.

In order to obtain the desired structure, first of all I discarded the rows of the dataset associated to the love emotion. Then I downloaded the well known *goEmotions dataset* [39], which can be used to fine-tune a model to classify a sentence in a discrete set of 27 emotions. I extracted from it the rows associated to the label *disgust* and an adequate number of rows (with respect to the current distribution represented in Figure 3.19) associated to the label neutral, and I appended them to the dataset described above, obtaining the resulting label distribution that can be seen in Figure 3.20. Finally, I created the Dataset object from the Huggingface Transformers library already described in Section 3.2.4, defining for each item in the dataset a input ids vector, an attention mask tensor and a label tensor that, in this case, specified the emotional label associated to the item.

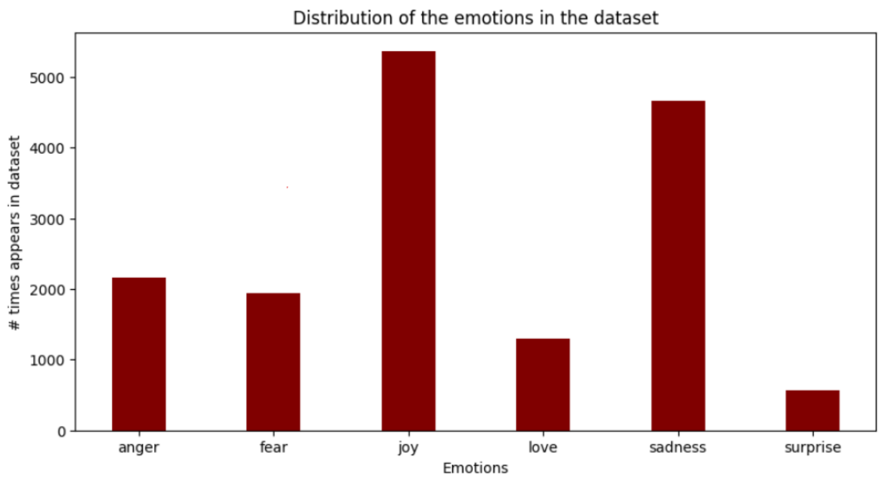


Figure 3.19: Distribution of emotions in starting dataset.

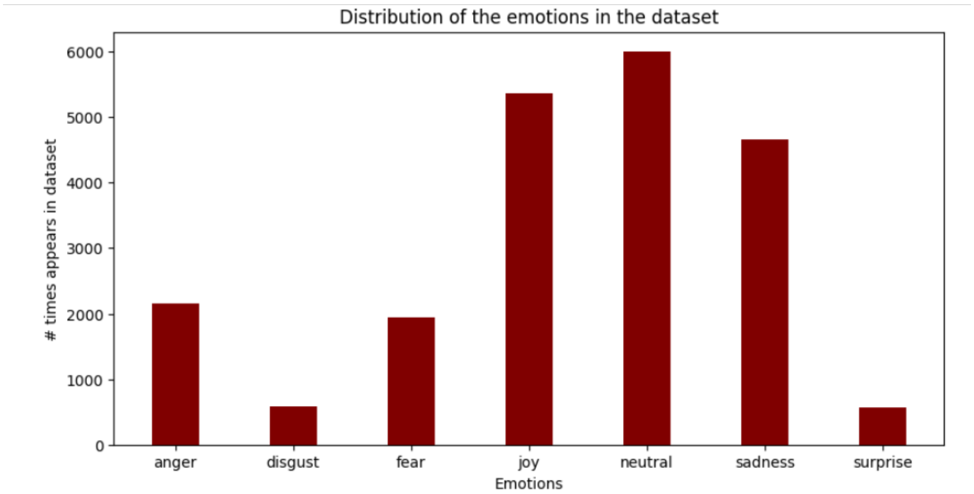


Figure 3.20: Distribution of emotions in final dataset.

Fine tuning

Once built the dataset, I instantiated the model, that was the base version of Bert with 140M parameters, and the tokenizer specifying the number of labels, in this case 7 (6 emotions plus the neutral one). The fine-tuning procedure I used was quite the same than the one described in Sec tion 3.2.4. In particular I used the Huggingface Trainer class with the following configuration:

- learning rate of $5 * 10^{-5}$,
- weight decay of 0.05,
- batch size of 16,
- Adamw as optimizer,
- linear learning rate scheduler.

Also in this case, the results are described in Section 5.

3.4. Speech To Text

Speech To Text (STT), also known as *Automatic Speech Recognition* (ASR), is the task to convert a human-like speech in textual form. It is a very important and delicate part of this project, since a bad quality transcription of the human interlocutor sentence would lead to bad response generation from the language model and to a confuse and meaningless dialogue. There were three necessary conditions to consider successfully accomplished this task:

- choose a model able to perform the STT task with high accuracy in order to pass as input to dialoGPT a context identical (or at least as close as possible) to what the human interlocutor said;
- perform the task in real time: in order to introduce as less latency as possible, the conversion of speech to text had to be performed live to be able to send the resulting sentence to the dialogue generation module as soon as the human ended to speak;
- remain offline: one of the constraints of the project was to be independent from any online API. For this reason, it was necessary to find a service that operates locally without using any external service.

I considered two different alternatives namely *Vosk* and *Nvidia Riva*.

3.4.1. Vosk

The first alternative I considered was *Vosk*, which is an offline open source speech recognition toolkit that supports more than 20 languages and dialects. It is available, for each language, both in a small version (around 40MB) that suits into lightweight devices (such as Raspberrys and smartphones) and a large version (around 2GB) for more accurate predictions.

It had mainly two problems:

- it didn't have a high accuracy with non-native English speakers, resulting in inaccurate results;
- it didn't detect autonomously punctuation, and in order to perform punctuation restoration task an additional model had to be loaded and used increasing dramatically the disk usage and the latency introduced.

3.4.2. Nvidia Riva

Nvidia Riva is a GPU-accelerated SDK for building speech AI application allowing to easily download, customize and fine-tune state-of-art models that deal with language as well as access their functionality through APIs. For what concern this section I was interested in the ASR service offered by Riva, but also on the TTS service that I will describe in details in Section 3.5.

Riva supports both data center and embedded architectures (this last one was my case, see Section 3.6) and since I didn't have particular requirements to satisfy and it wasn't the core of my project, I chose to exploit the already existing pre-trained models that I deployed, through the platform, locally. In particular, I used *ngc*, a hub that provides containers, models, scripts and in general solutions to facilitate the research in deep-learning field, to download a "starting kit" and to easily and quickly deploy locally the services. More specifically the files contained in the package were:

- *config.sh* configuration file: it allowed to set, among other parameters, which kind of models should be deployed locally. In my case, I selected the ones for STT and TTS tasks;
- *riva_init.sh* script: it can be used to initialize the service by downloading and deploying locally the models chosen through the configuration file;
- *riva_start.sh* script: it can be used to start the service, which means to run a server in a docker container on a port specified in the configuration file. It exposes APIs that can be called from a client;
- *riva_stop.sh* script: it can be used to stop the docker container on which the server runs.

3.4.3. ASR models

Conventional ASR systems are made by three main components:

- an *acoustic model* to predict the relationship between sub-phonemes, which are speech units, and an audio signal;
- a language model to assign probabilities to sequences of words;
- a *pronunciation model* to map phonemes to words.

These elements are used to determine the most likely hypothesis to decode the detected sounds [85]. Riva allows to easily and quickly deploy more advanced models: I will briefly describe the main features of few of them and I will finally report the one I selected as well as the main motivations of my choice.

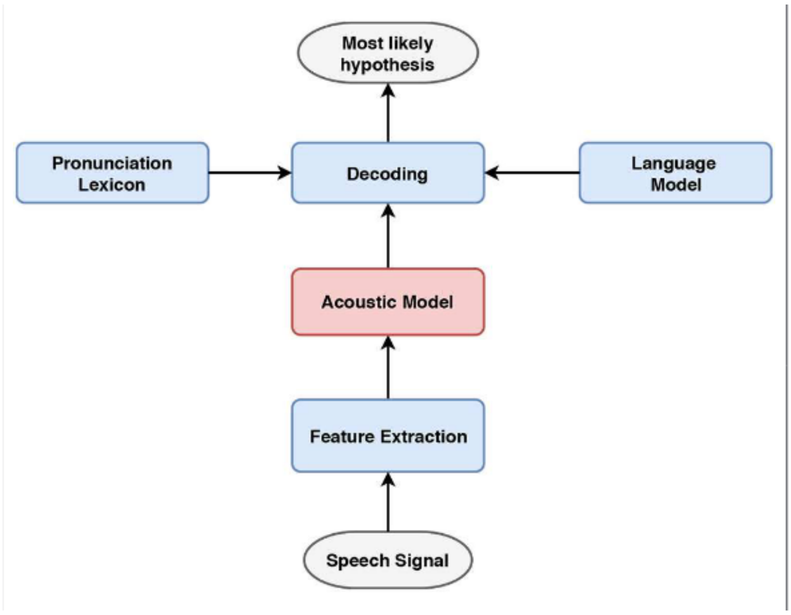


Figure 3.21: Conventional ASR systems structure.

Jasper

Jasper [59] is a family of ASR models which main feature is to replace the acoustic and pronunciation models with convolutional neural networks. Its architecture is made by blocks: Jasper BxR has B blocks each one with R sub-blocks, each one applying a sequence of operations (1-D convolution, batch normalization, ReLU and dropout) and uses residual connections to directly connect each block with the last sub-block. It is a *CTC model* since is trained using *Connectionist Temporal Classification* loss (CTC loss), which is, without going into details, a loss used for sequence modeling problems, such as speech recognition, where the input and the outputs have different size and the sequences should be aligned in some ways. This architecture was designed to facilitate and speed-up GPU inference.

QuartzNet and Citrinet

QuartzNet [57] architecture is based on the Jasper one but replacing the 1D convolutions with 1D time-channel separable convolutions and using larger kernels. The consequence is that the performances are similar to the ones of Jasper but with a fewer number of parameters. A QuartzNet BxR is made by B blocks each one containing R sub-blocks each one composed by a 1D separable convolution, batch normalization, ReLU and dropout.

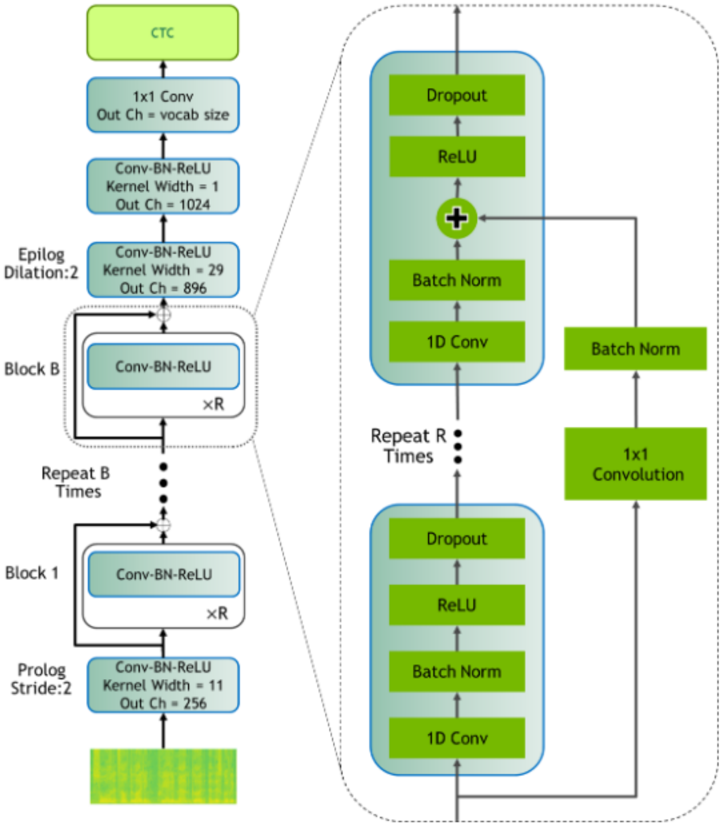


Figure 3.22: Jasper architecture.

Citrinet [64] is a CTC model that adds to the QuartzNet architecture with 1D separable convolution the *squeeze-and-excite-mechanism*, which is a unit able to improve the representational power of a network.

Conformer-CTC

Conformer-CTC is based on the *Conformer* architecture [48] which has the goal to combine convolution and self-attention to exploit the best of both the systems, since convolution is better to detect local correlations, while self-attention to understand global interactions. The Conformer audio encoder processes the input with a convolution layer and then with a sequence of conformer blocks that replace the transformer blocks. Each conformer block is composed by a feed-forward module, a multi-head self-attention module, a convolution module and another feed-forward module. The model has three different dimensions (small, medium, large) with different number of parameters and obtained with different depth of the network and different number of attention heads.

Conformer-CTC encoder [50] has the same structure described above with the only dif-

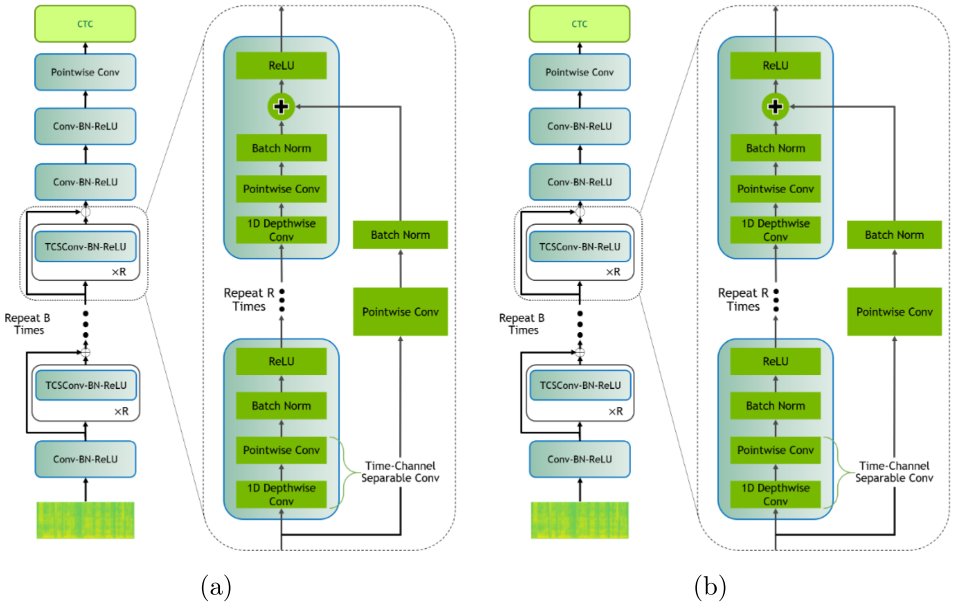


Figure 3.23: (a) QuartzNet architecture. (b) Citrinet architecture.

ference that is trained with a CTC loss.

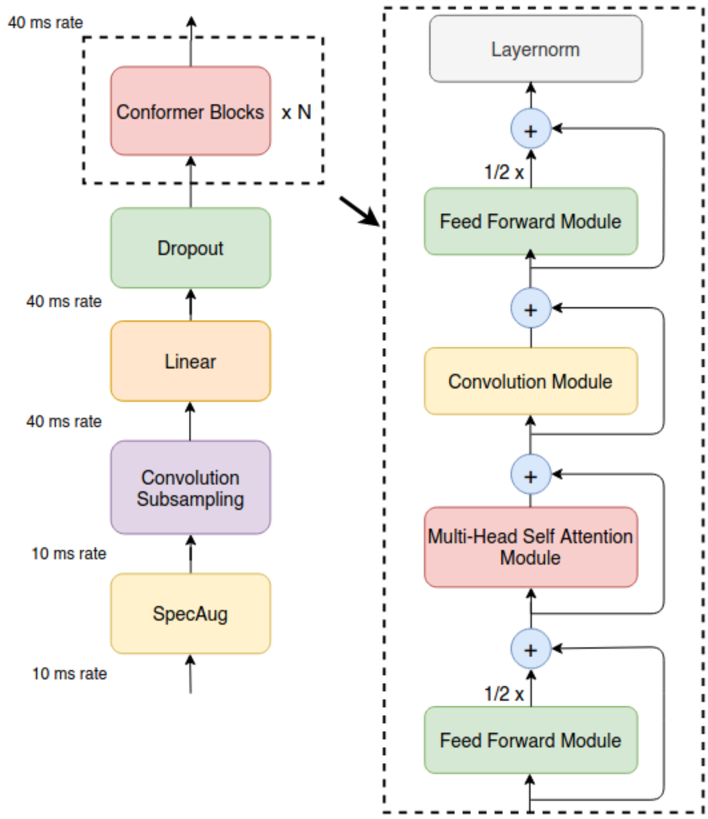


Figure 3.24: Conformer decoder architecture.

3.4.4. Model selection

To summarize, I had several possibilities to perform speech-to-text offline: at the end I chose to use the Conformer-CTC model through Nvidia Riva. The reasons were quite simple:

- first of all the Riva SDK is optimized for embedded systems and in particular for the Nvidia GPU I have used (see 3.6);
- I empirically tested both Vosk and Conformer-CTC models and the outcome from a qualitative point of view was extremely different, showing a greater ability of the second one to correctly convert to text the speech of non-native speakers (Vosk most of the times confused some words with others having a similar pronounce);
- I found in the literature comparisons between several ASR models and it turned out that Conformer-CTC out-performed on most of the evaluation datasets used. A summary of this can be seen in figure 3.25

Dataset	Vosk Aspire	Vosk Daanzu	Facebook RASR	Facebook Wav2Vec2.0	Nvidia Citrinet	Nvidia Conformer-CTC
Librispeech test-clean	11.72	7.08	3.30	2.6	2.78	**2.26**
Tedlium test	11.23	8.25	5.96	6.3	5.61	**4.89**
Google commands	46.76	**11.64**	20.06	24.1	28.15	19.77
Non-native speech	57.92	33.31	26.99	29.6	28.78	**24.22**
Children speech	20.29	9.90	6.17	**5.5**	6.85	**5.52**
Podcasts	19.85	21.21	15.06	17.0	14.82	**13.79**
Callcenter bot	17.20	19.22	14.55	22.5	**12.85**	13.96
Callcenter 1	53.98	52.97	42.82	46.7	36.05	**32.57**
Callcenter 2	33.82	43.02	30.41	36.9	29.40	**27.82**
Callcenter 3	35.86	52.80	32.98	40.9	29.78	**27.44**

Figure 3.25: Comparison of the performances of some ASR models on different datasets.

3.5. Text To Speech

Text To Speech (TTS) can be considered the opposite of STT and is the conversion of text into phonemes. I implemented this module in my project to produce the speech of the robot as soon as dialoGPT generate the response to the sentence of the human interlocutor, in order to simulate a real spoken conversation between two people. For this purpose I used two elements:

- a language model locally deployed through Nvidia Riva 3.4.2 to perform *speech synthesis*, which allows to append to the text necessary information to produce the sound, such as the spoken language, the sex of the speaker, the pronunciation...
- the python library *sounddevice* that allows to play *Numpy* arrays containing audio signals.

This module was a marginal part of my project and for this reason I decided to spend not much time in the choice of the model (Riva generally uses by default state-of-art models) and in the optimization and customization of the sound.

3.6. Hardware architecture

Since my project has to be integrated inside a wider research as an embedded system, it was necessary to have an hardware with mainly 2 features:

- it had to be compact enough to be mounted inside the robot structure in a comfortable way, and possibly not too heavy;
- it had to be enough powerful to be able to manage big and complex deep learning models in an effective way.

The choice was the 32 GB version *NVIDIA Jetson AGX Orin* module 3.26 which is compact, has lots of connectors, and is designed to be integrated on robots and powerful machines. The main features can be summarized as follows [15]:

- NVIDIA Ampere architecture GPU with 1792 NVIDIA CUDA cores and 56 Tensor cores;
- 8-core Arm Cortex-A78AE v8.2 64-bit CPU 2MB L2 + 4MB L3;
- deep-learning and vision accelerators;
- 32 GB 256-bit LPDDR5;
- 64GB storage (extensible with SD);
- 3 possible power settings (15W, 20W, 50W).

Figure 3.26: Jetson AGX Orin module.

4 | Software Architecture

In this chapter I will describe the software architecture of my project starting from a brief description of the main features of *ROS* which is the framework used to develop my modules and proceeding with a detailed description of how I implemented every functionality following the natural flow of information inside the system.

4.1. ROS

In this section I will briefly describe what is ROS and its newer version ROS2, which is the framework used in this project.

4.1.1. ROS main concepts

Robot Operating System [24] is an open-source middleware, which can be considered a bridge between the application and the low-level details, used to develop robots and presented in 2009. The provided services are for example the abstraction of the hardware level and the management of the very low-level processes, package management, implementation of commonly-used functionalities and message passing between different processes. The main features and advantages of ROS are:

- it is a distributed framework,
- it is language independent,
- testing is fast and easy,
- it facilitates the reuse of code,
- it is scalable,
- it has a huge community.

Since my purpose is just to give a very high level overview of the environment in which I developed my system I will not enter deeply in the details of ROS, but I will just outline the main arguments to introduce (only) the concepts and the terminology that I will use in this chapter:

- *nodes*: executable units, each one represents an independent process that can exchange data with other nodes;
- *topics*: channels of communication between nodes associated to a specific message type and implementing the publisher/subscriber paradigm. One or more nodes can publish messages (of the proper type) on a topic and each node subscribed to that topic will receive the message;
- *messages*: structured pieces of information that can be sent over topics in order to exchange data between nodes. Lots of predefined and standard messages already exist in ROS, but new ones with custom structures can be easily defined;
- *services*: remote functions called following the client/server paradigm. A node can synchronously call a service, which receives a request with a specific structure, performs a computation and gives back a response with a specific structure;
- *master*: it is the entity, unique in a system, that is responsible for the registration and the naming of nodes and services allowing the nodes to find and contact each others.

4.1.2. ROS2

In this project I have used ROS2 which is the newer ROS version published in 2020. There are lots of differences between the two versions, including low-level aspects such as the building system, but I will just outline few high-level but relevant changes [23]:

- ROS2 has standardized the way to write nodes with the OOP paradigm, by introducing the convention of creating a class that inherits from the *Node* class all the necessary methods to manage its lifecycle (such as the creation of publishers, subscribers, services...);
- the centralized system with a master is suppressed, since all the nodes are autonomously able to discover the others;
- the services are not necessarily synchronous, that means that the caller doesn't have to wait for the response.

4.2. General architecture

Before describing the implementation details of each node inside the system it is useful to have a general overview of its structure to understand at a high level how the different modules communicate and are related to each others. The architecture can be visualized in figure 4.1, and proceeding from left to right can be seen:

- SST module to convert human speech to text;
- an interface that connects my system (that manages the verbal interaction) to the one already present (controlling the physical interaction);
- a module in charge of generating the response of the robot in a conversational setting and of extracting the emotional state of the interlocutor exploiting a dedicated service;
- TTS module to convert the robot textual response to speech.

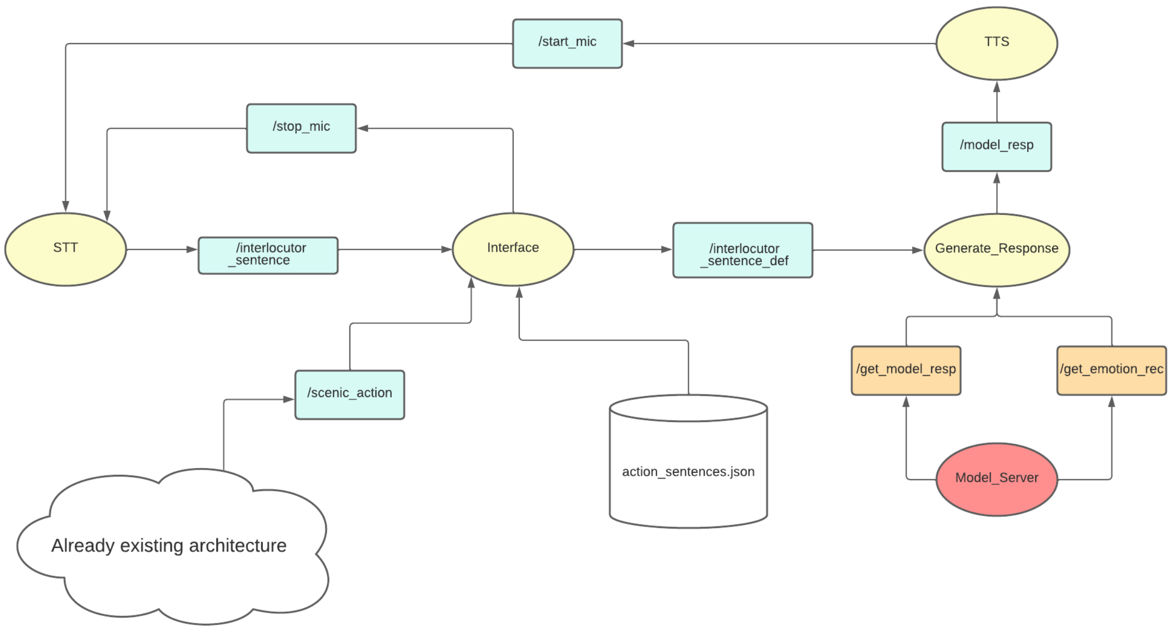


Figure 4.1: System architecture.

4.3. STT Node

This node is responsible for performing the Speech-To-Text task by exploiting the services exposed by the Riva server (see 3.4.2) through the library *riva.client*.

4.3.1. Basic operations

This node is subscribed to the topics:

- *start_mic*: on this topic are received ping messages to communicate to this node that can start to listen and to wait for a sentence from the human interlocutor;
- *stop_mic*: on this topic are received ping messages to communicate to this node to stop the microphone and to ignore any audio signal from the external;

and publishes on the topics:

- *interlocutor_sentence*: on this topic are published the sentences converted from speech to textual strings;
- *mic_is_listening*: on this topic are published ping messages to communicate to other nodes that the microphone is active and is currently listening to a speech, which means that the human agent has started to verbally interact with the robot.

The main activity of this node is just to loop waiting for the human interlocutor for starting speak, and then to use the Riva Client library, after having set few parameters such as the language (which is English) and the sample rate (16KHZ), to convert the English speech into text with punctuation. When the final string is detected it is returned a python dictionary with the structure:

```
{
    transcript: String,
    confidence: Float
}
```

, where the *transcript* is the string containing the detected sentence from the interlocutor through the microphone, whereas *confidence* is the logarithm of the confidence, which is an estimate of how confident the system is that the recognition is correct. Since generally the confidence score is between 0.0 and 1.0, in this case after the logarithm is applied is between *-inf* and 0.0, which means that the closer is to 0.0, the more is confident of the correctness.

At this point the main problems I had to face were three:

- the sound produced by the speakers generated a return in the microphone producing a loop, since the robot kept answering to itself;
- the environmental noise was sometimes detected by the microphone producing unexpected responses of the robot;

- every short pause of the interlocutor speech was interpreted by the Riva system as the end of its sentence and the result was immediately published to be processed by next nodes, generating strange responses.

4.3.2. Solutions

For what concern the return in the microphone, I decided to make a simple assumption: after the human interlocutor ends his sentence the microphone has to ignore any external audio stimulus from the environment until the robot finishes to produce the response. I just created a class inheriting from the Riva class *MicrophoneStream* (that has the task to manage the buffer of bytes detected through the microphone) adding methods to start and stop the listening from the microphone. Then, I simply used them to stop the microphone to prevent it to listen after the node published a detected sentence, and to start the microphone when a message on the topic start_listening, that is sent from another node after the robot response is produced through the speakers, is received.

To reduce the environmental noise I opted for a more "concrete" solution: I used a directional headset microphone and I exploited its closeness to the interlocutor mouth modulating two parameters:

- the volume of the microphone that I could keep very low to isolate it from external sounds;
- the value of the minimum confidence to accept the resulting detected sentence: every string with a lower confidence was rejected and considered noise from the environment.

The third problem was the most complex to be managed. The basic idea was to create a temporal window of n seconds (where n is a parameters that can be configured) to allow the interlocutor to keep speaking after the end of a sentence was detected due to a physiological and natural break. For this purpose I implemented a kind of custom timer that periodically checks its state and when reaches its end it triggers the final sentence publication process. In particular, when a string, that is considered as final, is detected by Riva, a new parallel thread handling the timer is run, and then there are two possibilities:

- the interlocutor doesn't speak again inside the configured temporal window: in this case the string is published as it is at the end of the timer;
- the interlocutor says something else: in this case the new detected string is appended inside a buffer, the timer is reset and when it finally ends a sentence obtained concatenating the strings in the buffer is published.

4.4. Response Interface Node

The aim of this node is basically to create a bridge between my project and the already existing system, by exchanging information and synchronizing the verbal reactions of the robots with the body ones. This integration process represent the weak point of the work also due to the difficulties met on a parallel research involving the body of the robot that made the coordination complex. For these reasons I made two assumptions:

- the data received from the existing system are correct and well structured,
- the physical reactions of the robot and the emotions expressed through the movements are always coherent with the context of the dialogue.

In particular, the desired goal was to develop a robot able to generate responses coherent both with the last sentence pronounced by the interlocutor (and generally with the context of the conversation) and with the emotions expressed through his gestures and expressions. Since a language model can obviously take as input textual data I decided to map any scenic action to a string that represents it (e.g., the action *sharing fear* can become "I am frightened"). It also represented a challenging and interesting field to test the flexibility and the generative power of the language models.

4.4.1. Basic operations

The node is subscribed to the topics:

- *scenic_action*: on this topic the modules developed in previous works by the colleague L.Farinelli [46] publish the scenic action of the human agent detected analyzing his facial expressions and body gestures;
- *interlocutor_sentence*: the messages in this topic are published by the STT node (3.4) and contain the last sentence of the human interlocutor converted in a string;
- *mic_is_listening*: the messages on this topics are ping sent by the STT node to communicate that the human interlocutor is speaking into the microphone;
- *start_mic*: the messages on this topics are ping sent to communicate that the model response has been generated through the speakers, so that the global state of this node is reset;

and publishes on the topics:

- *interlocutor_sentence_def*: on this topics are published the final strings that will be processed by the GPT-2 language model to generate the robot responses;

- *stop_mic*: the messages on this topic are ping sent to the STT node to force the stop of the microphone.

The objective of this node is to receive and synchronize the data coming from the STT node and from the already present system, in order to merge the information and create a string that well represents the stimuli coming from both the verbal and the movement dimension of the interlocutor.

The messages containing the detected scenic action of the human agent are received with a high rate (about 1 message every 0.5 seconds) and have the structure:

```
{
    action: Int,
    label: String
}
```

, where *action* is an Integer between 1 and 14 and represents the index of the scenic action (the order and the consequent indexes had been chosen a priori) and *label* is a textual representation of the action (mainly useful for debug purposes). Since most of the messages contain the *none* action, which means that nothing is detected, and since it can happen that isolated scenic actions are received due to both the high rate of sending and noisy detections, I adopted the following strategy.

Every time a message containing the scenic action A is received the node updates a buffer, which is a FIFO queue, of dimension N and checks if it contains at least X instances of A, where both N and X are parameters that can be configured. If the answer is *true* it is assumed that the scenic action A is effectively happening and it is not related to a mistake of the detection system, and there are two possible scenarios:

- the microphone is not active which means that the interlocutor is not speaking: the node sends a signal to stop the microphone, converts A into a string and publishes it;

- the microphone is active, which means that the human agent is speaking and performing a scenic action at the same time: the action A is saved inside the node waiting for the sentence S converted by the STT node; when it is received the action A is converted into a string, concatenated with S and published.

4.4.2. Conversion of scenic actions to strings

The mapping between scenic actions and strings is managed through a json file with the following structure:

```
{
    "attack": {
        "verbal_reaction": true,
        "sentence": "aaa",
        "response_time": 3
    },
    "intimidate": {
        "verbal_reaction": false,
        "sentence": "",
        "response_time": 3
    },
    "disappointment": {
        "verbal_reaction": true,
        "sentence": "I am disappointed.",
        "response_time": 3
    },
    "grudge": {
        "verbal_reaction": false,
        "sentence": "",
        "response_time": 3
    },
    "scolding": {
        "verbal_reaction": false,
        "sentence": "",
        "response_time": 3
    },
    "sharing sadness": {
        "verbal_reaction": true,
        "sentence": "I am sad.",
        "response_time": 3
    },
    "sharing fear": {
        "verbal_reaction": true,
```

```
        "sentence": "I am frightened.",
        "response_time": 3
    },
    "surprise": {
        "verbal_reaction": true,
        "sentence": "I am surprised.",
        "response_time": 3
    },
    "sharing happiness": {
        "verbal_reaction": true,
        "sentence": "I am happy!",
        "response_time": 3
    },
    "happy person": {
        "verbal_reaction": true,
        "sentence": "I am happy!",
        "response_time": 3
    },
    "satisfaction": {
        "verbal_reaction": true,
        "sentence": "I am satisfied.",
        "response_time": 3
    },
    "disbelief": {
        "verbal_reaction": true,
        "sentence": "I can't believe to it.",
        "response_time": 3
    },
    "astonishment": {
        "verbal_reaction": true,
        "sentence": "I am astonished.",
        "response_time": 3
    },
    "running away": {
        "verbal_reaction": false,
        "sentence": "",
        "response_time": 3
```

```
    },
    "none": {
        "verbal_reaction": false,
        "sentence": "",
        "response_time": -1
    }
}
```

Each element stands for a scenic action and contains three information:

- *verbal_reaction*: it is a boolean and communicates if a verbal reaction is necessary, since for some scenic actions a purely physical reaction could be enough without the need of speaking;
- *sentence*: it is the textual representation of the scenic action;
- *reaction_time*: it is an integer that represents the number of received messages of a specific scenic action to consider it happening and trigger the response generation process.

4.5. Response generation

This node receives the final sentence created by the interface 4.4 and mainly performs two actions:

- compute the response using the fine-tuned dialoGPT model described in 3.2.4;
- extract the emotion expressed from the sentence using the emotion recognition Bert model described in 3.3.2.

Both these operations are executed calling the remote functions exposed by ROS services (the node behaves as a client). The reason of this choice was that I wanted to make the system more scalable than possible, allowing other nodes that could be eventually added in future (for any reason) to perform the tasks of text generation and emotion recognition just calling the services.

4.5.1. Basic operations

The node is subscribed to the topic:

- *interlocutor_sentence_def*: on this topic the node receives the final strings from which compute the robot response;

publishes on the topics:

- *model_response*: on this topic are published the responses generated through the GPT-2 model;
- *emotion_sent_analysis*: on this topic are published the emotions extracted from the interlocutor sentences through the Bert emotion recognition model;

and behaves as client for the services:

- *compute_response*: the input is the interlocutor sentence and the output is the response from GPT-2;
- *sentiment_analysis*: the input is the interlocutor sentence and the output is the emotion extracted from it.

The operations executed by this node are quite simple and can be summarized as follows: it receives the interlocutor sentence, it creates two requests containing it and invokes the two services to compute the robot response and to extract the emotion, it publishes the results on two topics.

4.5.2. Language model Services

The service that performs emotion recognition has a very simple structure: it just downloads the proper tokenizer and loads from the disk the fine-tuned Bert model that classifies a sentence in 1 of the 6 Ekman emotions plus the neutral one, and uses it to extract an emotion from the string. The response contains an integer between 0 and 6 that represents an emotion with the following mapping:

```
{
0: 'anger',
1: 'fear',
2: 'joy',
3: 'love',
4: 'sadness',
5: 'surprise',
6: 'neutral'
}
```

For what concerns the service that manages the text generation, it downloads the tokenizer and loads from the disk the dialoGPT model fine-tuned on the ED Dataset. It keeps locally a buffer of length 3 that represents the context given to the model as input: every time the service is called the queue is updated inserting the new sentence passed in the request and the input context is created by concatenating the strings in the buffer. The response is produced passing to the *generate* method of the model the following parameters:

- *context_tokenized*: a tensor containing the tokens representing the input context;
- *max_new_tokens* = 56: it means that at most 56 new tokens are generated;
- *top_p* = 0.9: it means that only the tokens associated to a probability $\geq$ than 90% are considered;
- *top_k* = 50: it means that only the most likely 50 tokens are considered;
- *do_sample* = true: it means that the next token is extracted from a probability distribution.

The reason of these choices is that the combination of top_p and top_k with sampling discards very low ranked tokens adding at the same time some dynamicity to the selection; moreover, after a lot of trials and errors the output generated with this configuration seemed more coherent and better from a qualitative point of view than others. Obviously, these considerations are completely subjective but some comparisons with the results obtained with other hyper-parameter values and with other decoding methods (such as greedy-search) are reported and analyzed in 5.

After the generation the response is decoded and returned to the client.

4.6. TTS Node

This is probably the simpler node of the architecture since its role is limited to convert the textual response generated by the dialoGPT model to speech and reproduce it through the speakers.

It is subscribed to the topic:

- *model_response*: on this topic the response generation node 4.5 publishes the sentences that represent the robot responses;

and publishes on the topic:

- *start_mic*: on this topic this node publishes the pings to communicate to the STT node that the robot response has been converted to speech and then the microphone can be activated to listen to the human interlocutor.

When a string is received the node makes a request to the Riva server to synthesize it specifying also other parameters such as the used language, the encoding type (in this case linear pcm), the sample rate and the kind of voice that should be used (male or female). Then, the response from the server is wrapped in a numpy array and the voice is reproduced through the python library sounddevice.

When this procedure is completed a ping to let the STT node open the microphone is sent.

4.7. Scenic action generator

Since it was impossible to properly test the integration of my system with the already existing architecture because of some problems encountered during a parallel thesis project on the robot body and movement, I implemented a simple node that acts like a console to simulate the sending of scenic action messages. It just loops waiting the insertion by the user of an integer between 1 and 14: when a command is received the number is mapped to a scenic action and a sequence of ROS messages of the proper type are sent over the topic *scenic_action*. This mapping between the integer inserted and the scenic action published is specified through a json file with the following structure:

```
    {
    "0": "attack",
    "1": "intimidate",
    "2": "disappointment",
    "3": "grudge",
    "4": "scolding",
    "5": "sharing sadness",
    "6": "sharing fear",
    "7": "surprise",
    "8": "sharing happiness",
    "9": "happy person",
    "10": "satisfaction",
    "11": "disbelief",
    "12": "astonishment",
    "13": "running away",
    "14": "none"
}
```

The scenic actions are then received and processed by the interface node 4.4 which is subscribed to the topic.

5 | Results

In this chapter, the main results of this project are reported along two different dimensions:

- quantitative dimension: the performance of the trained models (dialoGPT for dialogue generation and Bert for emotion recognition) is presented in terms of metrics in order to justify the reasons why I selected them;
- qualitative dimension: some examples of conversations between a human interlocutor and the robot are presented, focusing on its ability to interact in a positive way manifesting empathy in different situations (happy, sad, chit-chat context...) and on the coherence of the generated responses. In my opinion these results are the most relevant because, even if the judgment is very personal and subjective, it is also true that there not exists a metric that is able to measure in a proper way the ability of a model to act adequately in an emotional and emphatic context, and it is better to let a human listener (or in this case reader) to evaluate the quality according to his sensitivity.

Finally, the outputs obtained by using different decoding methods and tuning the hyper-parameters at inference time are presented.

5.1. Quantitative results

The core of this project was to find out which was the most suitable language model able to perform in a conversational context with satisfying qualitative results according to the constraint of keeping everything saved locally. I have analyzed the features of several models and the designed choice finally was the medium version of dialoGPT for the reasons widely debated in Section 3.2. In order to choose the best configuration for the project settings, I have made lots of trials varying several parameters:

- learning rate: in the range $[5 * 10^{-6}, 0.1]$;
- batch size: 16 and 32 (I chose to not try larger batch sizes because of the limited available disk space);

- weight decay: in the range [0.1, 0.9];
- I have trained the model using different optimizers, namely AdamW and SGD with momentum;
- I have trained the model using different learning rate schedulers, namely linear and exponential.

Finally to select the best model I evaluated them on the test sets of three different datasets:

- ED dataset [69]the one used to fine-tune the model presented in Section 3.2.3),
- *Daily Dialogue Dataset* [60] which contains conversations grounded in everyday life situations,
- *Personachat Dataset* [86] which contains dialogues as well as information about the profiles of the interlocutors.

The table below summarizes these tests, reporting the performance of three models on the datasets mentioned above: the final dialoGPT medium (trained with AdamW optimizer, learning rate $5 * 10^{-5}$ and weight decay 0.05), dialoGPT small (trained with AdamW optimizer, learning rate $5 * 10^{-5}$ and weight decay 0.05) which is interesting to outline the differences introduced just varying the model complexity and the number of parameters and dialoGPT medium trained using SGD+Momentum as optimizer with learning rate 0.01 and momentum 0.9.

	DialoGPT M	**DialoGPT S**	**DialoGPT M (SGD+Momentum)**
ED	7.7069	8.9093	8.6915
Daily	22.7876	27.6765	23.1663
Persona	13.3124	16.9814	13.3094

Table 5.1: Performances of three different models on three different datasets using perplexity as metric. The models are: dialoGPT medium trained with AdamW, dialoGPT medium trained with SGD + momentum and dialoGPT small trained with AdamWdaily. The datasets are: ED Dataset, Daily Dialog Dataset and Personachat Dataset.

I report here only few examples of models I have tested in my trials, the ones that I considered particularly interesting, but the main observation that can be done is that the final selected model performs well on the same dataset it was trained on, but has good results also on different datasets, manifesting its flexibility and its ability to generalize. The training details of this model are showed below: Figure 5.1 shows the decreasing trend of the training error while Figure 5.2 represents the evaluation process using the perplexity metric, that reaches the minimum value around the training step 1000 (after 2 epochs) and then starts to increase.

For what concerns the emotion recognition task, since it didn't represent the core functionality of this project I just looked for a model with reasonably high performances: the Bert fine-tuned 3.3.2 accuracy on the test dataset was around 85%, and it was considered enough for the purpose.

Figure 5.1: Training error plot.

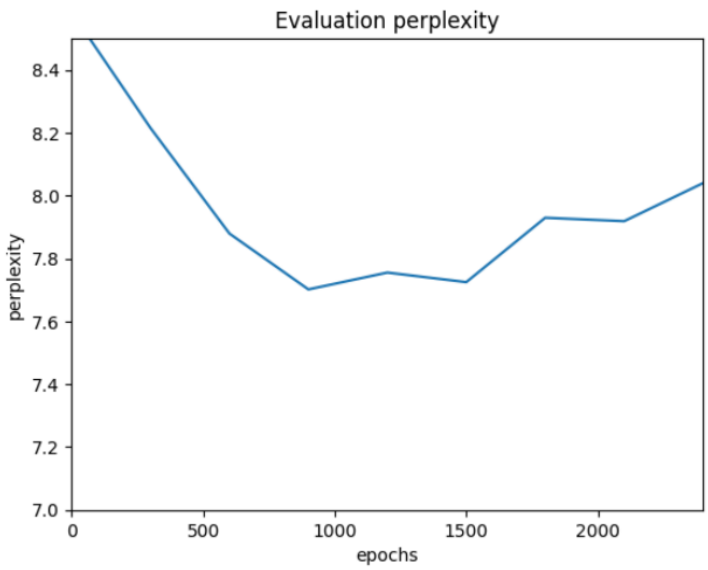


Figure 5.2: Evaluation perplexity plot.

5.2. Generated dialogues

Probably the best way to evaluate the ability of the robot to interact with a human showing empathy and awareness of the emotional context is to try to speak with it. For this reason, below are reported some examples of real short dialogues I had, in which its conversational ability are tested in several situations (the sentences in uppercase are the scenic action converted to strings as mentioned in Section 4.4): the results are quite satisfying since the responses are generally coherent with both the context and with the emotional state suggested by the input sequence even if sometimes it happens that it generates strange and unexpected sentences. Anyway, in the improvisational field in which this research is grounded a good actor could take advantage of this fact to create funny situations.

Example 1

User: Oh, my God, this is one of the best days of my life.
Model: I know right. What happened?
User: I got a promotion at work this morning.
Model: Congratulations! What did you get?
User: A better position and the higher salary
Model: That's good to hear! What's the next step of your career?
User: I don't know. I ought to become a manager, but the limit is the sky.
Model: Wow, I've never heard of someone doing that, that's so interesting!
User: I AM SATISFIED.
Model: I'm happy you are satisfied, I would be too!
User: Anyway, what is your job?
Model: I work in a grocery store

Example 2

User: I am reading the Harry Potter books, and I love them.
Model: I used to love them, but they are getting boring now.
User: Really? Why do you think that?
Model: I am a fan of books but the series was disappointing for me
User: Oh, I'm so sorry. What is your favorite series, then?

Model: Oh! I really liked Death Note and School Days.
User: I like that note *(asr module error, it should have been: "I like death note").* Who is your favorite character of it?
Model: The main one was Koyotee.
User: That's cool, you see. That's cool. You seem an expert to me.
Model: I am, he is my favorite character.
User: Anyway, do you have any other hobbies?
Model: No, i don't really.
User: you don't like doing sports, for instance?,
Model: I do not like sports either, but it is fun to do and I do like to make myself some food, so I don't feel sad.
User: Do you like to cook, then?
Model: Yes, I do, the cook is really nice.
User: What is your favorite dish to cook?
Model: Anything really, but my favorite meal is chicken

Example 3

User: I feel so sad today. I'm so depressed.
Model: What's making you depressed?
User: I AM SAD.I feel like I don't have any friends.
Model: Have you considered joining a sports club?
User: I don't know, I'm so shy.
Model: Have you tried joining an athletic club?
User: Why?
Model: That is usually a good way to feel more social and fit.
User: I AM FRIGHTENED.I think other people will not like me.
Model: Why?
User: I don't know, I'm just not so confident about myself.
Model: That is understandable. I'm sure if you put yourself out there, you will find your way out.
User: Thank you, you are a nice friend.
Model: You're welcome. I hope it works out for you too.

Example 4

User: Oh, my God, there is a spider in the living room.
Model: I hate spiders. They always try to get in my house.
User: I AM FRIGHTENED.I don't know what to do now.
Model: Oh my god, you need to exterminate it!
User: How can I do it?
Model: You are probably the only one who can.
User: I AM DISAPPOINTED. Why you can't help me?
Model: Have you tried praying, reading the books, and meditating.
User: I can't believe to it.Oh, I can't believe you are joking right now.
Model: I don't believe in angels either. You should try not to be so pessimistic.
User: Okay, okay, maybe I'm exaggerating. I will try to get rid of the spider now.
Model: You are brave, good luck with that!
User: Anyway, what is your biggest fear?
Model: The dark, but mostly the cold.
User: Why are you afraid of the cold?
Model: I think there's a lot of negative feedback there.

Example 5

User: Hello, what's up?
Model: what are you doing today?
User: I'm just relaxing and reading a book.
Model: i love relaxing, what book are you reading?
User: It's the 2nd one of Game of Thrones.
Model: thats what i am reading too! i have read all the books, are you going to watch all the tv shows?
User: Yes, sure. Have you already watched them?
Model: yup, i read the books like last year. i finished book 3 and finished the series
User: Oh, that's cool. And what is your favorite part of it?
Model: i really like the ending, so good.
User: That's amazing. I can't stand to see it.
Model: i really hope the ending is good for you!

Example 6

User: This is an amazing day. I have won the lottery.
Model: Woohoo! I've been gambling a lot too. What's your favorite lottery ticket?
User: I don't know, I just bought a random one.
Model: Nice! You should probably check the lottery numbers and win!
User: Yes, I already did it. am so proud of myself.
Model: Well that's a good start.
User: I AM HAPPY! What. should I do now with the money?
Model: Just keep working.
User: But what?
Model: How about you spend some on something you love.
User: That's a good advice. I would love to go to Japan for a journey.
Model: That sounds like an exciting time
User: Do you like Japan?
Model: No but I like sushi.
User: Haha. Great answer. I love sushi, too.
Model: I love seafood, love sushi, love anything with seaweed in it.
User: You should come to Japan with me, then.

Example 7

User: Hello.
Model: Hi. How are you doing?
User: I'm fine. I'm just working on my thesis.
Model: Are you in school?
User: University. I will get my master degree soon. I hope so.
Model: How exciting! You should be proud.
User: I am. What about you? Do you attend university?
Model: I went to school but transferred to a new job.
User: I AM SURPRISED. What kind of job?
Model: I have been doing retail for about 7 years now.
User: Wow, it seems you have a lot of experience in that field.
Model: I do too.
User: What do you like about your job?

Model: I like helping people.
User: That's cool. I would like to have a job like that.
Model: Are you looking for a job?
User: Yes, I am.
Model: What type of job are you looking for?
User: I will get a degree in computer science. Saw something related to it.
Model: Oh, wow! Congratulations! Is it something you are passionate about?
User: Yes, sure I love it.

Example 8

User: Hi. how are you?
Model: I'm great. How are you?
User: I think you are a little bit low.
Model: What are you going to do?
User: I'm trying to talk with you.
Model: Maybe you could have a coffee or something to calm down.
User: Yeah, you're right. I will take my coffee soon.
Model: Awesome. Coffee's the best. What are you going to do?
User: I'm expecting to. Play a little bit with the robot.
Model: Sounds like fun! Good luck. You'll be fine if you use good timing.
User: Would you like to play with a robot?
Model: That would be fun!
User: Are you able to talk with a robot?
Model: I guess you could. I'm afraid of talking though.
User: Why are you afraid of a robot? You shouldn't be afraid of a robot.
Model: I guess I just don't like talking too much.
User: So the problem is not that the robot is talking.
Model: It's just you and me.
User: Yes, but I was wondering about talking. Talking with the robot.
Model: We can't talk with people if they don't have hands.
User: What do you mean?
Model: What happens if you talk to someone whose hands are full, but it is not a person?
User: It may respond. If it is a robot, it may respond.

Example 9

User: I'm reading the Game of Thrones books, and I am loving them.
Model: I've never read them! I do love books though.
User: Yes, I love fantasy books.
Model: Me too! I can't read any non fantasy at the moment.
User: Haha, same. What is your favorite fantasy book?
Model: I love Night Lords.
User: What is it about?
Model: It's about a guy who is the night lord and the people who live in the night lord's castle are his prisoners.
User: Wow, Sounds school. Do you recommend it to me?
Model: It's like a classic teenage fantasy book with lots of action and lots of wacky humour.
User: I don't know. I think it's not for me. What about Lord of the rings? Have you read it?
Model: No, I haven't. I know it's in the library though!
User: You should try to give it a chance. It is a very classic fantasy.
Model: I'll check it out then!

5.3. Decoding methods comparisons

In order to choose the most proper decoding method and tune the hyper-parameters I once again tried lots of different combinations to find out which one was the most appropriate to generate dialogues with a human interlocutor. A useful experiment was to give the same input to the same model configured with different parameters at inference time to be able to evaluate the main differences.

Greedy search:

User: What are you doing today?
Model: I'm going to go to the beach.
User: Wow sounds funny! Are you with some friends?
Model: I am with a few friends.
User: Can I come?

Model: Sure! I'll bring the beer.

Sampling with top_k and top_p:

User: What are you doing today?
Model: I am in a hotel room doing a bunch of laundry, what about you?
User: that sounds boring, why are you doing it in an hotel room?
Model: For the warm feelings of the warm air!
User: don't you have warm air at home?
Model: No, it's too dry, my friend!

Beam search with 4 beams:

User: What are you doing today?
Model: I'm doing a lot of reading.
User: Sounds funny! What are you reading?
Model: I'm reading a lot of books.
User: what is your favourite?
Model: I like to read a lot.

Beam search with 4 beams and sampling:

User: What are you doing today?
Model: I'm going to the gym.
User: Wow sounds funny! I love working out.
Model: I'm glad you're enjoying it!
User: Can I come with you?
Model: I'm sure you can!

The finally selected decoded strategy was sampling with top_k and top_p (3.2.5) and the reason is quite simple: one of the main features that a good interlocutor should have in order to maintain a positive conversation is the ability to surprise and engage the other person, even if sometimes strange and not completely coherent sentences can be created. On the other hand, greedy methods just select the next token associated to the highest probability given a sequence and the result is that for the same input always the same output is generated, while beam search strategies have the problem to tend to predict boring and predictable sentences.

6 | Conclusions and future developments

This thesis work was inserted in a broader research aimed at building a robot able to interact with a human actor manifesting and processing emotions both from a verbal and non-verbal point of view in an improvisational context. For what concern my project, the goal was to implement the ability of having a conversation with a human agent generating responses that manifested empathy, coherence with the overall context and that were able to surprise and entertain the interlocutor creating a positive relationship with him.

The results that I reached are:

- the robot is able to detect when the interlocutor begins to speak and to coordinate and synchronize the activities of generating the responses and reproducing them through the speakers;
- the robot is able to dialogue with a human interlocutor in an improvisational context generating responses that show a general knowledge of the situation;
- the robot is able to manifest empathy generating responses that are adequate from an emotional point of view with the state and the mood of the interlocutor;
- the verbal dimension is synchronized and connected with the non-verbal one: the emotions are extracted from the interlocutor sentences in order to generate proper movement reactions and at the same way the scenic actions detected from the human actor movements are used by the language modules to generate adequate verbal responses.

At the same time, there are some weak points:

- the responses generated by the robot sometimes appear strange, not perfectly coherent with the last interlocutor sentence or even fall in contradictions. For sure small improvements could stil be done, but these problems are mainly due to the limited power of the model used and not to the way I trained it: dialoGPT medium has indeed interesting features, but the performance can not be compared to the ones of larger and more complex models. The reason of this choice was that the constraint I had to satisfy was to keep everything locally and then I had to select a model that could fit into the embedded system.

- the speech recognition system is not able to analyze the semantic of the detected content and interprets every small pause as the end of the interlocutor speech with the consequence that the robot generates responses to not-complete sentences. I have tried to introduce a delay in the robot response generation but the result is that the conversation sometimes appears "non-natural" and a bit fragmented.
- the synchronization of the two parts (the dialogue manager that I have developed and the general robot management, which considers gestures) is not fully optimized.

6.1. Future developments

6.1.1. Test other models and datasets

In order to try to improve the performance and the quality of the outputs the first simple idea could be to try other language models to generate the robot responses. The reasons why I selected the dialoGPT model was that it reached good results and was widely used in the existing literature about emphatic chat-bots, but, at the same time, the research on language models is growing exponentially in these last years and new solutions can be found. An example of interesting trial could be the *T5-base* model which was published by Google in 2019: it has a total of 220M parameters and a dimension that could be compared to the one of the medium version of GPT-2, but differently from it has an encoder-decoder architecture and does not use only decoder transformer blocks.

For what concerns the dataset, I used the Emphatetic Dialogue dataset because the main goal was to implement on the robot the ability to apparently understand the emotional state of the interlocutor and properly react to it, but several other solutions can be useful for the improvisational theatre context in which this project is grounded. Few examples are:

- *Cornell Movie Dialogs Corpus*: it contains over 200K conversations from over 600 films;
- *Personachat dataset*: it contains over 160K conversations from couple of people with information about their persona to create more customized dialogues and maintain a state of their personality during the whole interaction;

- *SPOLIN corpus*: it contains *yes-and* exchanges extracted from improv podcasts.

These are only examples (lots of others useful sets of data can be found) that could be used to pre-train and/or fine-tune the selected language model in order to improve the performances.

6.1.2. Improve the coordination of the modules

The main problem about the integration of my project inside the already existing architecture was that a parallel thesis work with the purpose of improving the selection of the next scenic action performed by the robot through machine learning techniques encountered some issues that made impossible to coordinate and synchronize the modules in the best way. For this reason the two parts have been implemented in a modular and independent way and coordinated only at the end: a future necessary development should be the integration of them in a coherent and homogeneous way, by keeping in consideration the overall design and concept of the project.

6.1.3. Create custom dataset

Probably the best way to try to integrate in an homogeneous and coherent way the modules that manage the physical movements and the modules that control the verbal reaction to the human agent would be to create a custom dataset in order to fine-tune the selected language model. Since the hypothetical testing environment is the improvisational theatre the best scenario would be to create a corpus of data taken from real improvisational exchanges and, in order to fulfill the complete integration of the parts, the sentences could be commented with information about the scenic actions performed by the actors. An example could be for example:

A: [scenic action performed by A] sentence1
B: [scenic action performed by B] sentence2
ecc...

where each scenic action label could be declared as a token of the model.

The only problem of this approach is that the creation of this custom dataset from scratch would be very time consuming since thousands of conversations would be needed in order to train properly the model.

Bibliography

[1] The Attention Mechanism from Scratch. https://machinelearningmastery.com/the-attention-mechanism-from-scratch/.

[2] Beyond Seq2Seq Architectures. http://chrome.ws.dei.polimi.it/images/d/d7/AN2DL_06_2223_BeyondSeq2Seq.pdf.

[3] How to generate text: using different decoding methods for language generation with Transformers. https://huggingface.co/blog/how-to-generate.

[4] 5 Reasons Emotions Are Important. https://www.verywellmind.com/the-purpose-of-emotions-2795181, .

[5] Emotion. https://www.britannica.com/science/emotion, .

[6] ENIAC Display: Computer Science Engineering at Michigan. https://www.britannica.com/art/theatre-art/Theatre-as-expression/.

[7] Evolution of Language Models. https://ai.plainenglish.io/evolution-of-language-models-cce8f6bf19a0.

[8] Understanding GRU Networks. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

[9] Glossary. https://huggingface.co/transformers/v2.11.0/glossary.html#token-type-ids, .

[10] Perplexity of fixed-length models. https://huggingface.co/docs/transformers/perplexity, .

[11] Trainer. https://huggingface.co/docs/transformers/main_classes/trainer, .

[12] Transformers. https://huggingface.co/docs/transformers/index, .

[13] The Illustrated GPT-2 (Visualizing Transformer Language Models). http://jalammar.github.io/illustrated-gpt2/.

[14] Improv Comedy Explained: 5 Tips for Doing Improv Comedy. https://www.masterclass.com/articles/improv-comedy-explained/.

[15] NVIDIA Jetson AGX Orin Series. https://www.nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf.

[16] Language Modelling. https://paperswithcode.com/task/language-modelling, .

[17] Language models and RNN. https://medium.com/@rachel_95942/language-models-and-rnn-c516fab9545b, .

[18] linguistic ambiguity. https://www.techtarget.com/whatis/definition/linguistic-ambiguity/.

[19] Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[20] Multi-turn chatbot project (3): GPT-2 chatbot with multi-turn generation settings. https://songstudio.info/tech/tech-35/#2.

[21] N-Gram Model. https://devopedia.org/n-gram-model#Jurafsky-and-Martin-2009.

[22] An Introduction to Recurrent Neural Networks and the Math That Powers Them. https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/.

[23] ROS1 vs ROS2, Practical Overview For ROS Developers. https://roboticsbackend.com/ros1-vs-ros2-practical-overview/#Why_ROS2_and_not_keep_ROS1, .

[24] ROS. http://wiki.ros.org/ROS/Introduction, .

[25] Understanding Encoder-Decoder Sequence to Sequence Model. https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346.

[26] Theatre as expression. https://cse.engin.umich.edu/about/beyster-building/eniac-display/.

[27] What is Improv? https://www.hideouttheatre.com/about/what-is-improv/.

[28] Word Embedding. http://chrome.ws.dei.polimi.it/images/2/2d/AN2DL_05_2223_WordEmbedding.pdf.

[29] J. ANGEL FERNANDEZ. Theatrebot: Studying emotion projection and emotion enrichment system for autonomous theatrical robot. 2016.

[30] J. M. Angel-Fernandez and A. Bonarini. Robots showing emotions: Emotion representation with no bio-inspired body. *Interaction Studies*, 17(3):408–437, 2016.

[31] J. M. Angel-Fernandez and A. Bonarini. Enriching robot's actions with affective movements. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 424–429. IEEE, 2017.

[32] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[33] L. Bonetti. Design and implementation of an actor robot for a theatrical play. 2021.

[34] B. Branch, P. Mirowski, and K. W. Mathewson. Collaborative storytelling with human actors and ai narrators. *arXiv preprint arXiv:2109.14728*, 2021.

[35] M. Carlson. Psychic polyphony. *Journal of Dramatic Theory and Criticism*, pages 35–48, 1986.

[36] C. Chiroli. Simulation of emotional behaviour in a robot. 2022.

[37] B. M. Cuff, S. J. Brown, L. Taylor, and D. J. Howat. Empathy: A review of the concept. *Emotion review*, 8(2):144–153, 2016.

[38] J. Decety, I. B.-A. Bartal, F. Uzefovsky, and A. Knafo-Noam. Empathy as a driver of prosocial behaviour: highly conserved neurobehavioural mechanisms across species. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 371(1686): 20150077, 2016.

[39] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi. Goemotions: A dataset of fine-grained emotions. *arXiv preprint arXiv:2005.00547*, 2020.

[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[41] R. Dunbar. Why only humans have language. 2009.

[42] R. I. Dunbar. The social brain hypothesis. *Evolutionary Anthropology: Issues, News, and Reviews: Issues, News, and Reviews*, 6(5):178–190, 1998.

[43] P. Ekman. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200, 1992.

[44] N. Evans and S. C. Levinson. The myth of language universals: Language diversity and its importance for cognitive science. *Behavioral and brain sciences*, 32(5):429–448, 2009.

[45] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

[46] L. Farinelli. Design and implementation of a multi-modal framework for scenic actions classification in autonomous actor-robot theatre improvisations. 2022.

[47] D. Francis and S. Hester. An invitation to ethnomethodology: Language, society and interaction. chapter 1, pages 1–13. Sage, 2004.

[48] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[49] V. Guljajeva. Synthetic books. In *10th International Conference on Digital and Interactive Arts*, pages 1–7, 2021.

[50] P. Guo, X. Chang, S. Watanabe, and L. Xie. Multi-speaker asr combining non-autoregressive conformer ctc and conditional speaker chain. *arXiv preprint arXiv:2106.08595*, 2021.

[51] S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346, 1990. ISSN 0167-2789. doi: https://doi.org/10.1016/0167-2789(90)90087-6. URL `https://www.sciencedirect.com/science/article/pii/0167278990900876`.

[52] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

[53] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[54] F. Ioannidou and V. Konstantikaki. Empathy and emotional intelligence: What is it really about? *International Journal of caring sciences*, 1(3):118, 2008.

[55] K. Johnstone. *Impro: Improvisation and the theatre.* Routledge, 2012.

[56] R. Ketai. Affect, mood, emotion, and feeling: Semantic considerations. *The American journal of psychiatry*, 1975.

[57] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang. Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6124–6128. IEEE, 2020.

[58] I. Leite, A. Pereira, S. Mascarenhas, C. Martinho, R. Prada, and A. Paiva. The influence of empathy in human–robot relations. *International journal of human-computer studies*, 71(3):250–260, 2013.

[59] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J. M. Cohen, H. Nguyen, and R. T. Gadde. Jasper: An end-to-end convolutional neural acoustic model. *arXiv preprint arXiv:1904.03288*, 2019.

[60] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*, 2017.

[61] Z. Lin, P. Xu, G. I. Winata, F. B. Siddique, Z. Liu, J. Shin, and P. Fung. Caire: An empathetic neural chatbot. *arXiv preprint arXiv:1907.12108*, 2019.

[62] D. B. Lindsley. Emotion. 1951.

[63] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

[64] S. Majumdar, J. Balam, O. Hrinchuk, V. Lavrukhin, V. Noroozi, and B. Ginsburg. Citrinet: Closing the gap between non-autoregressive and autoregressive end-to-end models for automatic speech recognition. *arXiv preprint arXiv:2104.01721*, 2021.

[65] J. D. Mayer. What is emotional intelligence? 2004.

[66] P. Nandwani and R. Verma. A review on sentiment analysis and emotion detection from text. *Social Network Analysis and Mining*, 11(1):81, 2021.

[67] R. Plutchik and H. Kellerman. *Theories of emotion*, volume 1. Academic Press, 2013.

[68] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[69] H. Rashkin, E. M. Smith, M. Li, and Y.-L. Boureau. Towards empathetic open-domain conversation models: A new benchmark and dataset. *arXiv preprint arXiv:1811.00207*, 2018.

[70] C. Reeck, D. R. Ames, and K. N. Ochsner. The social regulation of emotion: An integrative, cross-disciplinary model. *Trends in cognitive sciences*, 20(1):47–63, 2016.

[71] R. Rosa, O. Dušek, T. Kocmi, D. Mareček, T. Musil, P. Schmidtová, D. Jurko, O. Bojar, D. Hrbek, D. Košťák, et al. Theaitre: Artificial intelligence to write a theatre play. *arXiv preprint arXiv:2006.14668*, 2020.

[72] J. A. Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161, 1980.

[73] E. Saravia, H.-C. T. Liu, Y.-H. Huang, J. Wu, and Y.-S. Chen. Carer: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3687–3697, 2018.

[74] G. Sheen. Carl: An empathetic chatbot.

[75] S. Tomkins. *Affect imagery consciousness: Volume I: The positive affects*. Springer publishing company, 1962.

[76] S. Tomkins. *Affect imagery consciousness: Volume II: The negative affects*. Springer publishing company, 1963.

[77] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950. ISSN 0026-4423. doi: 10.1093/mind/LIX.236.433. URL https://doi.org/10.1093/mind/LIX.236.433.

[78] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[79] Wikipedia contributors. Stage — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Stage&oldid=1121210196, 2022. [Online; accessed 19-February-2023].

[80] Wikipedia contributors. Univac i — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=UNIVAC_I&oldid=1119595474, 2022. [Online; accessed 19-February-2023].

[81] Wikipedia contributors. Neural network — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Neural_network&oldid=1134823540, 2023. [Online; accessed 28-February-2023].

[82] Wikipedia contributors. Gpt-2 — Wikipedia, the free encyclopedia. https://en.

wikipedia.org/w/index.php?title=GPT-2&oldid=1142023473, 2023. [Online; accessed 2-March-2023].

[83] Wikipedia contributors. Improvisational theatre — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Improvisational_theatre&oldid=1139314428, 2023. [Online; accessed 19-February-2023].

[84] Wikipedia contributors. Language model — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Language_model&oldid=1141710774, 2023. [Online; accessed 27-February-2023].

[85] J. Wu, E. Yılmaz, M. Zhang, H. Li, and K. C. Tan. Deep spiking neural networks for large vocabulary automatic speech recognition. *Frontiers in neuroscience*, 14:199, 2020.

[86] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*, 2018.

[87] Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*, 2019.

List of Figures

List of Tables

Acknowledgements

Here you might want to acknowledge someone.