



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

EXECUTIVE SUMMARY OF THE THESIS

Marketing infrastructures and tools designed for international expansions

LAUREA MAGISTRALE IN COMPUTER SCIENCE AND ENGINEERING - INGEGNERIA INFORMATICA

Author: MOHSEN POURSHIRAZI

Advisor: PROF. GIANPAOLO CUGOLA

Academic year: 2021-2022

1. Introduction

ProntoPro is a marketplace that connects people and small businesses in need of a service with artisans and professional workers offering that service. To make it happen, ProntoPro needs to find the people in need of a service as well as the people capable of delivering such services. In this project, we have focused on building and maintaining tools to help ProntoPro persuade people capable of delivering services to use ProntoPro as their marketing tool.

This project is conducted to help ProntoPro with its goal of international expansion, focusing on European countries by providing proper infrastructures and tools to decide on when, where, how and who to contact in order to find potential customers.

Here you can find the description of some terms used in the project:

Professionals: People specialized in providing a particular service who possess the required prerequisites and qualifications. Such as teachers, plumbers, movers, etc.

Merchants: The professionals that have joined ProntoPro's platform and are paying for its services.

Prospects: Professionals that could potentially become future merchants.

This project is aimed to find as much information as possible regarding prospects by crawling the web and other internet resources and automatically extracting and classifying relevant information.

2. Project overview

The project is divided into three main parts:

1. Gathering the data from the world wide web using automated tools.
2. Preprocessing the data and ingesting it into ProntoPro database.
3. Providing features and functionalities to use the data from the database.

Figure 1 shows a high level abstraction of the workflow adopted to achieve the ProntoPro objective.

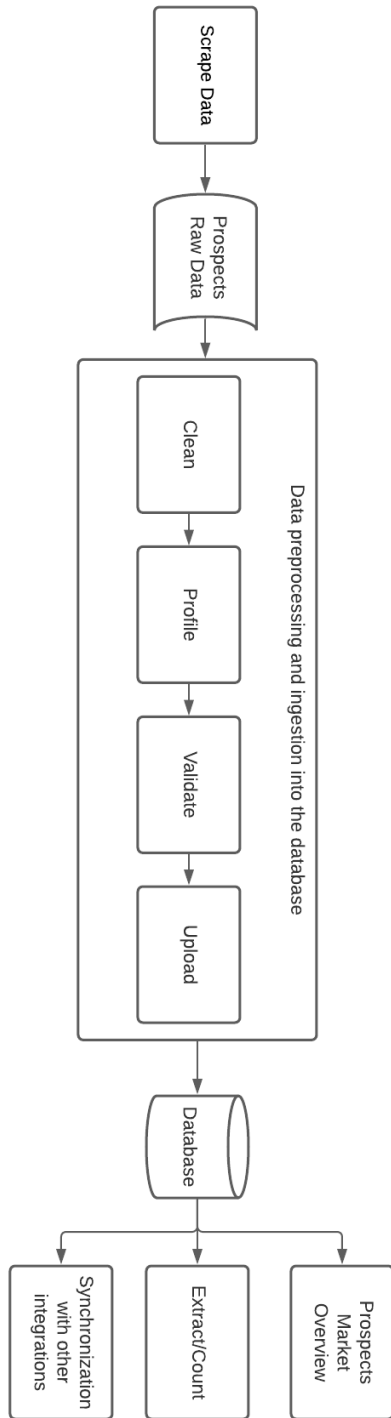


Figure 1: High level workflow of the solution

2.1. Gathering the data

This task is carried out by designing and implementing tools called web scrapers [2] which automatically extract data from various web sites.

For each of the targeted web sites, a customized web scraper has been created using Python as the programming language.

Below, you can find the libraries and frameworks we used in this process:

1. BeautifulSoup
2. Selenium
3. LXML
4. Python Requests

We have targeted certain sources to extract information about prospects regarding their occupation, location and ways to contact them.

2.2. Data preprocessing and ingestion into the database

Taking advantage of the Python libraries and frameworks called "pandas" [1], "re", "phonenumbers", "difflib" and "requests", logical inconsistencies are prevented and the extracted data is reformatted to match the structure of our designed database. Also the external taxonomy of professions and localities are mapped to the internal taxonomy of ProntoPro.

Moreover, Contact information of each prospect is validated with the help of third party services like debounce.io and customer.io for validating emails and phone numbers respectively.

Finally, prospects with duplicated contact information are merged together while prospects with no contact information are discarded before being pushed to the database.

2.3. Using the extracted data and application features

A general overview of all the records in the database is created using Tableau with a variety of customizable filters and graphical representations.

A software application is designed to offer clients with the following features through a graphical user interface:

- **Uploading** scraped prospect information.
- **Counting** prospects with optional filters.
- **Extracting** prospects with optional filters and the possibility to push the extracted prospects to a CRM tool called customer.io. This feature automatically flags the extracted prospects as "active" in order to prevent them from appearing again in the next extraction.
- **Synchronization** with other integrations

in order to update the prospects statuses properly in cases they have (un)subscribed from ProntoPro or requested to never be contacted by ProntoPro again. (This feature is only available to admins)

An example of the graphical user interface is given in the figure 2.

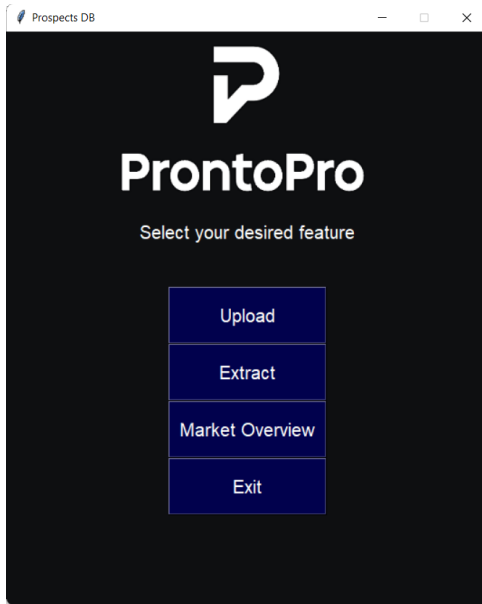


Figure 2: Main menu

3. Conclusions

With having more than 5.5 million prospects by the end of this project, it has been observed that the effect of such infrastructures and tools which are designed to be used intuitively will result in more informed decisions on markets activation and more than 1.9 million distinct emails, 5.8 million phone numbers, and 2.5 million company URLs have proved to be extremely useful when taking actions based on those decisions.

References

- [1] Wes Mckinney. pandas: a foundational python library for data analysis and statistics. *Python High Performance Science Computer*, 01 2011.
- [2] Vidhi Singrodia, Anirban Mitra, and Subrata Paul. A review on web scrapping and its applications. In *2019 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6. IEEE, 2019.



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Marketing Infrastructures and Tools Designed For International Expansions

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE AND ENGINEERING - INGEGNERIA IN-
FORMATICA

Author: **Mohsen Pourshirazi**

Student ID: 10632568
Advisor: Prof. Gianpaolo Cugola
Academic Year: 2021-22

Abstract

Lacking proper information and insights about potential customers might cause negative results while companies are expanding internationally. This project aimed to design and implement a system in which the relevant data denoting potential customer information is gathered, processed, and stored in a database later to be utilized. The data acquisition consists by multiple pieces of independent software commonly known as web scrapers or web crawlers. Then the data is processed in order to be compliant with the designed database structure alongside being mapped to the company's internal taxonomy before being ingested into the database. This project has been conducted within ProntoPro for internal company needs in order to provide proper insights about different markets for specific service-related businesses and locations that help ProntoPro with the process of making the right decisions on where, when and how to activate those markets. The final outcome of this project was an application capable of building a general overview of markets and extracting potential customer information through a graphical user interface.

Keywords: Web scraping, Python, Marketing, Expansion, Data gathering and pre-processing

Sommario

La conoscenza del mercato e dei suoi potenziali clienti è fondamentale per ogni azienda che intenda espandersi internazionalmente e la mancanza di informazioni precise e puntuali può portare al fallimento dell'azione di internazionalizzazione. Questa tesi mira alla progettazione e implementazione di un sistema in cui i dati riguardo potenziali clienti di un nuovo mercato vengono raccolti, elaborati e archiviati in un database per essere successivamente utilizzati. Il processo di acquisizione dei dati è realizzato attraverso l'uso di più software indipendenti, comunemente noti come web scraper o web crawler. Successivamente i dati vengono elaborati per essere conformi a quanto previsto dalla struttura del database in uso e mappati alla tassonomia interna dell'azienda per poi essere effettivamente inseriti nel database medesimo. Il presente lavoro è stato condotto all'interno di ProntoPro, il maggiore marketplace per servizi locali d'Italia, per esigenze interne all'azienda, al fine di supportare il processo decisionale che porterà ProntoPro ad espandersi in nuovi mercati esteri, aiutando a definire quali offerte di servizio attivare nei vari mercati di sbocco. Il risultato finale di questo lavoro è un'applicazione in grado di fornire una panoramica generale dei mercati ed estrarre le informazioni sui potenziali clienti di ProntoPro attraverso un'interfaccia grafica di facile utilizzo.

Parole chiave: Web scraping, Python, Marketing, Espansione, Raccolta dati e pre-elaborazione

Contents

Abstract	i
Sommario	iii
Contents	v
Introduction	1
1 Database structure and design	5
2 Gathering the data	9
2.1 Web scraping	9
2.1.1 Data Sources	12
2.2 Legal aspects of scraping	14
2.3 Legality of Web Scraping	15
3 Data preprocessing and ingestion into the database	17
3.1 Data cleaning (Cleaner)	17
3.2 Taxonomy mapping (Profiler)	21
3.2.1 Profiling localities	23
3.2.2 Profiling professions	26
3.3 Data Validation (Data pipeline)	28
3.3.1 Company URL checker	28
3.3.2 Email Debouncer	28
3.3.3 Email scraper	29
3.3.4 Email guesser	29
4 Using the extracted data and application features	31
4.1 Prospects market overview	31
4.2 Count and extract modules	32

4.3 Synchronization with other integrations	33
4.4 User interface	35
5 Conclusion and future work	37
Bibliography	39
A Appendix A	41
List of Figures	43

Introduction

ProntoPro is Italy's largest local service marketplace. It helps millions of customers to find a local service professional: covering over 600 services, ranging from wedding photographers over plumbers to guitar instructors. Within a few years, ProntoPro plans to make it possible that booking a plumber or a painter online will become as effortless as buying a book. Since 2019 it is in the process of international expansion, focusing on continental Europe.

Even though ProntoPro is experiencing an advanced phase of growth, the startup agility still guides their processes and decisions. ProntoPro has always been supported by successful investors and entrepreneurs. Being a part of Immobiliare network lets them develop innovative solutions without typical start-up risks. Their employees come from all over the world - there are more than 16 different nationalities at ProntoPro.

In this project we are going to define some terminology and clarifications as follows:

- **Professionals:** People specialized in providing a particular service who possess the required prerequisites and qualifications. Such as teachers, plumbers, movers, etc.
- **Merchants:** The professionals that have joined ProntoPro's platform and are paying for its services.
- **Prospects:** Professionals that could potentially become future merchants.

ProntoPro's marketing team includes two major sub-teams that were mostly involved in this project:

SEO: This team is responsible for making the company's website to appear more often in the SERP (Search Engine Results Pages) when people search for certain services. In other words, they are responsible to provide higher demand as well as supply.

ProAcquisition: This team is responsible for finding and persuading prospects to join the platform by running classical acquisition and growth channels, such as Email marketing, SMS, job boards, LinkedIn, influencer Collaborations and others. In other words, they are responsible for providing the supply.

This project is mainly affecting the ProAcquisition team by providing them with sufficient and reliable data. Finding legal and reliable contact information and establishing a

system to give people at ProntoPro an overall perspective over the potential markets is one of the main goals of the project.

Expanding to different markets on an international level requires enough information in order to make good estimations and forecasts. Also, the possibility of reaching out to potential customers is essential. As one can imagine, the world wide web provides a wide range of facts and data sources established by humans [20], and if a business can realize how to make good use of it, as a result, a higher revenue is expected. The business model of ProntoPro is defined in a way that paying customers are the people who possess the necessary skills to offer different services such as "Electricians", "Plumbers". You can find the full list of all services covered by ProntoPro at Appendix A.

Building automated tools for extracting and storing relevant data from the web is the initial step of this project, however there are multiple challenges along the way. As any other company, ProntoPro has its own taxonomy to address different things such as customer classification and geographical location for the sake of organizational matters. And since the data gathered from external sources have a high chance of being misaligned with these taxonomies, the mapping between data points is one of the major challenges. Also designing, maintaining or replacing the tools integrated in the project is very important. The reasons may vary from changes regarding architecture or organization structure of the whole company, to matters of efficiency, developing user needs, etc.

Figure 1, illustrates a high level abstraction of how the project is constructed. In the following, you can find brief descriptions of what happens in each process.

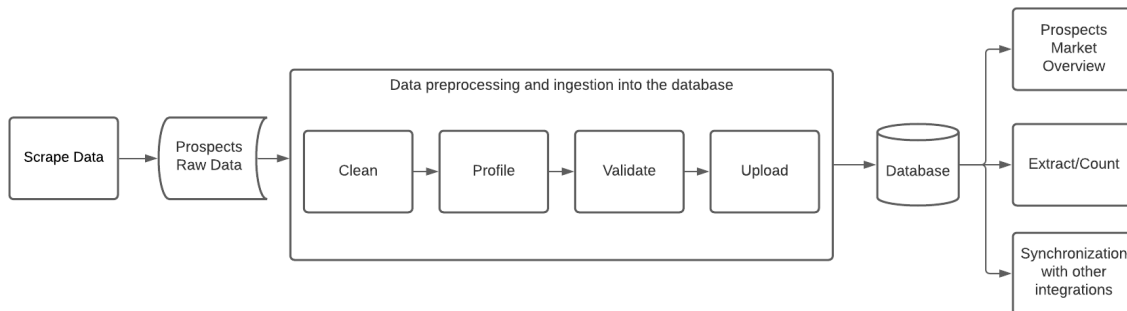


Figure 1: High level workflow of the solution

- **Scrape data:** This is where the process of scraping the web and extracting the raw data takes place.
- **Data preprocessing and ingestion to the database:**
 - **Clean:** Cleaning the scraped data to be compliant with our database structures while excluding logical inconsistencies which is done by the cleaner mod-

ule.

- **Profile:** Mapping taxonomies of external sources to the internal taxonomy of ProntoPro. The module responsible for this process is called profiler.
 - **Validate:** Making sure the data regarding the contact information is reliable. Actions are sequentially placed in a module called data pipeline to carry out this process.
 - **Upload:** Managing duplications and the process of uploading the profiled and validated data to the database.
- **Prospect market overview:** A general overview of all the prospects stored in the database presented graphically in certain dashboards alongside customizable filters for clients' needs.
 - **Extract/Count:** Getting insights about certain prospects and extracting the required information to contact them properly.
 - **Synchronization with other integrations:** Synchronization process to guarantee concurrency between databases of different departments.

1 | Database structure and design

Before initiating the data gathering process, we need to start with the architecture of our database. The most straightforward option would be using a traditional tabular database (also known as relational database). Since it is also the approach which almost all the other departments of ProntoPro have selected, we decided to take advantage of the existing subscriptions and infrastructures for hosting our database. As one can also contemplate, integration with other departments is necessary, especially regarding the task of tracking and monitoring the prospects who have been converted into merchants. The tool used for the process of setting up and maintaining the database is DBeaver, a free multi-platform database tool for developers, database administrators, analysts and all people who need to work with databases. It also supports all popular databases such as: MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, Teradata, Firebird, Apache Hive, Phoenix, Presto, etc.

In Figure 1.1, we can see the ER (Entity Relationship) diagram representing the connection between the tables and their corresponding columns.

The database consists of three layers:

First layer is the "prospect" table which is the main table of our database, all the relevant information about the prospects that might come handy to the ProAcquisition team is stored here. Each prospect can have multiple contact points of different types, which leads us to the next layer.

The second layer, is the contact layer which consists of three tables:

- contact_email
- contact_phone
- contact_company_url

Each of these contact tables represent the potential way to contact a prospect. Taking contact_email table as an example: each record is associated with an email address presumably belonging to a prospect.

There are four columns that contain timestamps in order to help us classify each of our contact points:

- **status_ready_at**: is the timestamp indicating the time that the prospect with this particular contact point is created.
- **status_active_at**: is the timestamp indicating the time that the prospect is extracted and most probably contacted by a member of ProAcquisition team. More information about this process is given in section 4.2.
- **status_invalid_at**: is the timestamp that express the time of a contact point being considered as unusable due to a reason which can be found in the invalid_reason column.
- **status_subscribed_at**: is the timestamp showing the time that the prospect associated with this contact point is subscribed to the ProntoPro platform. More details in section 4.3

The last layer is the activity layer. Each of the contact tables are connected to an activity table that in short stores the history of how each record of the corresponding contact table is used. Each time a prospect is extracted with specifying an activity, an activity is created in this table for the related contact point. As explained before, each contact point can be classified into four different categories. We call these categories, "status" in order to avoid any misunderstandings with the categories in the ProntoPro's taxonomy. (e.g., "Home Improvement" is a category consisting of different professions)

Below, we have listed all the statuses that a contact point can be classified into:

- **Ready**: The contact is newly uploaded to the database and is ready to be used.
- **Active**: The contact has been reached out to through a ProAcquisition channel. (pending)
- **Invalid**: The contact is unusable due to one of the following reasons:
 - The prospect does not want to be contacted by ProntoPro. (black listed)
 - The contact is unusable. (unreachable)
- **Subscribed**: The prospect has already subscribed to ProntoPro. (merchant)

Each prospect can be flagged based on its contacts; as an example, if a prospect associated to one contact point classified as "ready", it can be deduced that this prospect is ready to be used.

Consequently, the status flags are derived from the contact tables and are not explicitly included in the prospect table, this is due to the fact that we can contact prospects through different channels and invalidity of one does not necessarily mean the invalidity of all the

others. For instance, if a prospect is unreachable via its company URL, maybe it is possible to contact it through its phone number(s). However, if a prospect is blacklisted or has already subscribed to the platform, it should be considered as invalid. In this project, a Python SQL toolkit and Object Relational Mapper (ORM) that enables application developers to have full power and flexibility of SQL, called "SQLAlchemy" has been used.

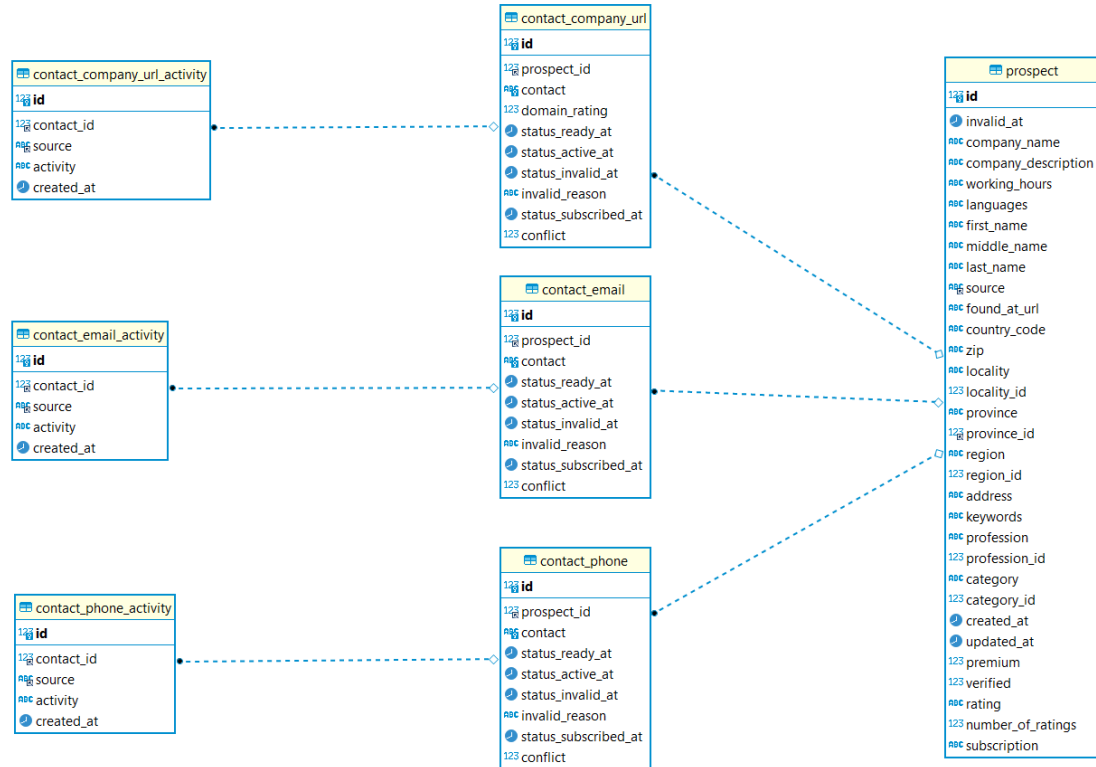


Figure 1.1: ER diagram

2 | Gathering the data

According to the projects' goals, we have to find prospects belonging to specific occupations in specific locations which usually goes by the term "market".

In the last decade, the drastic growth of data accessibility through the world wide web opens a window for companies to grasp the opportunity to reach out to customers in a faster and more practical manner [12]. Practical in the way that the selectivity and relevancy of services offered to the right people at the right moment would makes it more likely for companies to increase the chance of being profitable through marketing target strategies. Thus, making it possible for companies to discover as much information as they can about their potential customers, in order to contact them regarding their interests. For example, by contacting a personal trainer in the city of Milan with an email in Italian offering them the possibility of finding clients though our platform will result in a higher conversion probability, or in other words, encouraging them to choose ProntoPro as their marketing tool.

With all that being said, finding good sources of information and collecting them will be the steppingstone in the process. However, manually browsing through websites and gathering useful data from them is not feasible. This led us to invest our time in automating this task using a technique called Web Scraping.

2.1. Web scraping

Web scraping (also known as Screen Scraping, Web Data Extraction, and Web Harvesting etc.) is the process of automatic extraction of data from the web, instead of doing a manual copy of it [20]. In the procedure of web scraping technique, we find meaningful data from the HTML source of websites to be extracted and later be stored into a central local database or spreadsheet. In our case, the data is structured into certain shapes and then stored in files with CSV (Comma-separated values) format.

Operation of web harvesting is performed by web scrapers which are software applications. This software can either be assembled in a customizable way for extracting data from particular web pages or can be of the kinds that are designed for a generic use, by

organizations.

Some web scraping procedures can be listed as HTTP programming, HTML parsers and DOM (Document Object Model) parsing which is generating an internal structure in memory that is a DOM document object containing all the information of an XML document, and the client applications get information of the original XML document by invoking methods on this document object; DOM Parser has a tree-based structure. The generated data is later used for retrieving or analytical tasks. It is a big advantage as it grants us with error-free data as well as saving our time in giving super quick results and most importantly for us, to find and stores all the data we need at once.

Web scraping is currently utilized on various fields including online price comparison, weather data monitoring, website change detection, web mashup, web research and web data integration.

Although some websites offer APIs to obtain required information, it is not always the case.

The prevalent practices in the developing of web data scrapers can be categorized in three main groups: [20]

1. **Libraries for general-purpose programming languages:** As mentioned briefly, method commonly used by bioinformaticians is composed of implementation of individual web data scrapers with the help of using a known programming language, such as Python, Perl, etc. Generally, third party libraries allow us to access the site with the implementation of client side of the HTTP, while the processing of the retrieved elements is done by utilizing the built-in string processing and manipulation functions like comparison of regular expression, tokenization and trimming. Third party packages can also grant us the possibility to use complicated type of parsing such as HTML tree construction or XPath matching.
2. **Frameworks:** While using general-purpose programming languages for creating scraper robots, we might encounter several limitations. Sometimes, different libraries are dependent on integrations to APIs for accessing the Web to retrieve useful information from HTML documents. Furthermore, robots are known to be delicate software, strongly affected by the alteration of HTML of extracted resources, thus forcing us to modify them frequently. For the software developed in compiled languages such as Java, instead of recompiling and redeploying the total application, one can refer to scraping agendas which might illustrate extra explanation. For example, Scrapy, a framework in Python, outlines agents as classes originated from "BaseSpider" class, and introducing a handful of "parsing" functions and "starting URLs" to be used for every web iteration. During the process of parsing Web pages

for extracting contents, using XPath expressions is inevitable. alternative agendas might give away domain-specific languages (DSL), designed for particular domains resulting in robots to be considered as peripheral and self-governing objects. For example, Web-Harvest which is a Web scraping agenda for Java programming language, in which the procedure of extraction is defined with XML and is contained of several pipelines, comprised of procedural commands, such as defining variables as well as loops and many other primitives.

3. **Desktop-based environments:** In this type of instruments, conception and maintenance of robots have been enabled by graphically designed environments. Normally, users navigate through the target web page using a browser and collaboratively select the elements which they desire to be extracted. The major issues regarding these types of tools, are their commercial regulations with limited API access which prevents us from embedding scrapers inside other software applications (which is our need) [8].

Classification of automated web scraping methods can be listed as follows: [7]

1. **Syntactic Web Scraping:**

In this method, information is found by parsing the HTML, CSS and other web languages.

Here you can find some of the common methodologies in this approach:

- Content Style Sheet selectors: characterization of illustrative aspects of HTML elements can be used as one of the ways to select and extract data.
- XPath selectors: similar to CSS selectors, XML can be employed to find the desired HTML nodes.
- URI patterns: finding web resources which respect the pattern of a certain regular expression. Unlike CSS or XPath which are used to find elements at document stage, URI patterns allow selection of credentials as well.
- Visual selectors: HTML nodes are associated with some properties to be visualized in a browser. Web designers usually use comparable visual properties to help with identifying elements.

2. **Semantic Web Scraping:**

The data extracted using syntactic web scraping techniques can be compared with resources from the semantic web in order to achieve elevated demonstration and purposes. To attain this, various structures such as Resource Description Framework (RDF) and the Web Ontology Language (OWL) can be utilized. However, the

practice of this method is less frequent.

3. Computer vision web-page analyzing:

Using machine learning and computer vision technologies, the process of identifying and extracting of data can be done by artificial intelligence agents that are designed to roughly observe and understand the websites, similar to humans that can later be related with CSS selectors [25]. As an example, we can reference "Diffbot", a service that crawls the entire public web and provides a searchable knowledge graph [4].

Developing customized scraper software for different websites can be accomplished using Python as the programming language. There is a wide variety of options to choose from in Python libraries and frameworks that can be utilized to build the proper web scrapers which can be used to gather the needed relevant information from different websites. Some of the more common and well-known ones are listed below:

- BeautifulSoup
- LXML
- MechanicalSoup
- Python Requests
- Scrapy
- Selenium
- Urllib

The tools used in this project are mostly BeautifulSoup, LXML, Python Requests, and Selenium.

Selenium: Selenium is a powerful tool for controlling web browsers through programs and performing browser automation.

Beautiful Soup: BeautifulSoup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree.

2.1.1. Data Sources

In order to create functional and useful scrapers, defining and studying good sources to be used as our targets is an important step.

By investigating similar service providing web pages, finding prospects who are eager to

use Internet as their marketing tool would be beneficial. Some of the sites that are used in this project are listed below:

- <https://www.google.com/maps>
- <https://www.yellowpages.com/>
- <https://www.dasoertliche.de/>
- <https://www.dastelefonbuch.de/>
- <https://www.doctoralia.es/>
- <https://doncontacto.com/>
- <https://www.gelbeseiten.de/>
- <https://guiademicroempresas.es/>
- <https://www.houzz.com/>
- <https://www.indexacompany.com/>
- <https://www.milanuncios.com/>
- <https://www.misterwhat.com/>
- <https://www.paginasamarillas.es/>
- <https://www.thesaurus.com/>
- <https://www.topdoctors.it/>
- <https://www.werkenntdenbesten.de/>

For each of the data sources, we have constructed a proper scraper. In order to navigate the page and find the needed data, selenium is used to simulate the behavior of a website user in the process of finding and loading all the intended result pages and then by taking advantage of beautiful soup, we can extract the data into their corresponding fields.

We can comprehend more details in the following scenario:

1. Open the main page of the web site for a given URL. (Selenium)
2. Select the search bar and fill in the text with the relevant query. (Selenium)
3. Click on the search button. (Selenium)
4. Scroll down to load all the needed data in the result page. (Selenium)

5. Extract data from the downloaded page and fill out the necessary fields in a structured dictionary. (Beautiful Soup)
6. Save the dictionary into a .CSV file which will be delivered to the upload module.

Or alternatively:

1. Create the proper URLs.
2. Load the pages. (Python Requests/LXML)
3. Extract the entire content of the each page. (Beautiful Soup)
4. Parse the HTML and fill out the necessary fields in a structured dictionary. (Beautiful Soup)
5. Save the dictionary into a .CSV file which will be delivered to the upload module.

Considering the fact that each of the targeted web sources has its own specific design and regularities which requires us to behave and interact differently with each of them, we establish unique scraper tools for each source.

Taking advantage of the empirical tests and historical results from the ProntoPro SEO team, the most effective method to find relevant results in most of the usual platforms and websites is searching for the combination of “business” followed by a “locality” (city name) in the target country language; as an example, for finding best results for electricians in Milan, the most promising query for good results would be “Elettricista Milano”. As expected, the SERP will contain a variety of valuable information which must be structured and classified in order to comply with the design of the database as well as our intended input and output.

2.2. Legal aspects of scraping

Discovering and exploiting large amounts of data from the web, is followed by some serious technical, legal and ethical challenges. Although much advancement can be seen in designing proper tools and technologies aimed at developing web scrapers [17], legal and ethical aspects of gathering data from the web are still to be found in "grey areas" [18]. Unlike some legal frameworks which are practiced to some extent, ethical problems are usually bound to be ignored. To mention some of these legal frameworks, we can point out the ones that have been applied in court cases, such as: breach of contract, copyright, trespass to chattels, trade secrets and illegal access and use of data. It is crucial for us to be mindful of legal and ethical problems that might be caused by web scraping in order to evade very costly lawsuits alongside any other potential damage to the reputation of

ProntoPro. It has been concluded that there exists an inherent paradox related to data in the web, which complicates the extraction and analysis of it from a legal point of view [12]. Web data was intended to be available to the public which has affected the business model of a lot of companies. For example, higher revenue for a website owner, caused from a wider user base accessing available data on their website due to this openness. However, this data is an important property for the owner of the website which needs to be protected, thus they prefer to consider this data as propriety, which means it is owned by the people or entities behind the website containing this data.

This ownership comes with some complications, as the website is owned by the owner which does not necessarily indicate the fact that generated data by the website users also belongs to the owner [5]. As of now, no web scraping legislation has been developed so far, this can be due to the novelty of the web harvesting process or the complications regarding the data ownership. Nonetheless, there is a set of legal frameworks and principles in other contexts that can be considered as guidelines. Law and ethics are distinct concepts; however, they are complementary to each other [16]. In the topic of web scraping, more ethical controversies exist than the legal ones [12].

2.3. Legality of Web Scraping

As stated previously, no direct legislation addresses web data extraction directly. This is the reason behind why people involved in such activities are not always certain of their actions.

In the following, you can find some detailed information about some of the fundamental legal theories applied to web scraping: In many cases in courts, the main focus has been on whether the "Terms of Service" of websites forbids web scraping or not. But violation of the "Terms of Service" alone, might not be enough to be considered basis for liability under the CFAA (The Computer Fraud and Abuse Act) [6].

Availability of access might be taken away and become authorized, in cases when the website owner sends a cease-and-desist letter to the entity responsible for crawling the website [3]. Even so, there has been a court that did not hold the web scraper liable under the CFAA, with only the cease-and-desist letter [23]. Some courts have noted that web harvesting publicly accessible data does not violate the CFAA [9].

Generally, users need to agree on "Terms of Service" (e.g., clicking on a checkbox), in order to be considered liable in case of violating them which may lead to a "breach of contract" on the website's user side [1]. Therefore, forbidding web scraping on the website does not necessarily mean the prohibition of web crawling from a legal point of view. Also, for the website to be successful on a breach of contract claim, it needs to prove that

they have encountered material damages as the result of crawlers violating their "Terms of Service".

Extracting and publishing data that have been explicitly copyrighted, might result in a "copyright infringement" case. Particularly in cases when scraped data is used for financial benefits [5]. But there would be no restriction from the copyright law, if the data is just being collected, and we have to keep in mind that the data does not necessarily belong to the website. Especially when it is generated by the users.

It has also been a topic in courts, whether inexistence of authorization mechanisms in websites, can be considered a permission to copy scraped data or not. One of these mechanisms is the robot's exclusion protocol or simply robots.txt protocol which contains all the instructions about what areas of the website should not be processed or scanned [19]. Nevertheless, in scenarios that the data on the website does not belong to the website owner, creating an implied license by the robots.txt file is not possible [2]

Damaging websites or web servers by overloading it, can result in holding the person who caused the damage to be held liable under the "trespass to chattels" theory [5]. However, this damage must be material and provable in courts for financial compensation [10], which is rarely the case [24].

Web users are not obliged to obey the instructions given in the robot.txt protocol by any legal restrictions, but by not doing so, web crawling might lead to unintended damage to the website users and owners (e.g., privacy concerns).

From an ethical point of view, upon publication of personal and sensitive data, users might have some expectations regarding the protection and privacy of their data. For instance, web crawling can be the tool used by security and law enforcements officials in order to identify individuals and prosecuting them [26]. Additionally, users publish certain data, assuming they will be used in a particular context, and when they are used for other purposes, this will raise concerns regarding their privacy and consent.

The process of gathering information should always be one of the most cautious processes, since finding and storing private information of people without their consent can be considered as an illegal act. Based on that reason in particular, the extraction logic for finding contact information in sources that requires us to carry on with the process of registering and creating a profile which can be later used to login and access the needed information has been limited to the company URL and the landing page in which we have found the prospect that can be later used as a contact point or as a clue on where to find the contact information. It is also necessary to keep track of the time when scraping emails, phone numbers or any other private information has taken place, since we need to reach out to those people and ask their consent to store their information in a certain time period.

3 | Data preprocessing and ingestion into the database

After the process of crawling the web, the acquired data must be processed in order to be useful for the company before being uploaded to the database.

3.1. Data cleaning (Cleaner)

In this module, we start by organizing the raw files provided by our scrapers in a compliant way to our database architecture, meaning we structure them into predefined columns with the correct values. In other words, missing columns are going to be added and extra ones are going to be discarded. Also, we need to enforce the insertion of some particular fields, such as source (clue on where the data was scraped), country code, locality, and at least one keyword related to professions, because all of these fields are necessary to create and upload a new prospect to the database.

Making use of a Python library called "pandas", which includes very rich data structures as well as functionalities to integrate projects with structured data sets usually used in statistics, finance and social sciences [15], appeared to be an interesting option for this task.

Having the data in pandas "dataframe" objects (2-dimensional labeled data structure with columns of potentially different types), enables the possibility to perform common data manipulations and analysis.

One of the responsibilities of the cleaner module is transforming the content of each different field to a clean version. Clean, in the sense that they would be relevant and usable for the processes we might apply over them later. To simplify and guarantee the process of cleaning, having a particular structure for the raw scraped data alongside specifying the data type of each column in the dataframe, is very helpful. Nevertheless, bear in mind that this process was partially done in the scraper modules, however the necessity of having a more advanced cleaner module while keeping the complexity of the scrapers limited to the task of extracting data, encouraged us to develop a separate module

for the cleaning process.

After introducing the intended data types for every field, we can utilize different strategies to clean and organize them.

For columns which were identified as type "string" (e.g., company name), we must find and remove all the emojis, as they are of no use to us and may cause issues related to other processes in the future.

In the following, you can find brief definitions which might aid us understand the situation better:

- **Unicode:** The unicode standard is an encoding of characters and texts which is done universally and is defined by Unicode Consortium (legally Unicode, Inc.). It offers the possibility to exchange data internationally which is one of the foundations of developing software globally [22].
- **Unicode block:** A unicode block is structured from ranges of adjacent Unicode characters. Each of these blocks usually is used to provide glyphs for at least one specific language or some generic application area, like mathematics.
- **Emoticons:** A unicode block that contains emojis (graphical representation of faces, hand gestures, etc.).
- **Regular expressions:** Regular expressions, also known as regex, regexp and rational expression, is a series of characters that identifies a particular search pattern.

By identifying the emoticons and using regular expressions to find and remove them from strings, we can accomplish the task mentioned above. We can use the "re" (regular expression) library in Python which provides all the operations required for finding certain patterns such as the following unicode blocks:

- "\U0001F600-\U0001F64F" for emoticons
- "\U0001F300-\U0001F5FF" for symbols & pictographs
- "\U0001F680-\U0001F6FF" for transport & map symbols
- "\U0001F1E0-\U0001F1FF" for flags (iOS)

By using the "re.compile" method, we can compile these unicode ranges into regular expression pattern objects, then replace character sequences that match these patterns with an empty string (practically, removing them), using the "re.replace" method. We can also use the same approach for discarding non-words. Luckily the non-word pattern ("W+") is already available in the Python "re" library. It is also helpful to delete all the stop-words from string type fields. An example for one such field is the address. However,

commonly, each country has its own language and the stop-words are going to be different correspondingly. For this reason, we have constructed certain dictionaries containing most of the stop-words for the following languages: English, Italian, German, Spanish, and French. Improving these dictionaries can lead to better results, especially in times when some processes for string matching are involved. We can see the necessity of this process in 3.2 "profiler" module since the chance of finding two strings to match will increase drastically when some stop words are discarded. Taking advantage of the features and functionalities included in a Python library called "textthero", a toolkit that facilitates working with text-based dataset which is designed to be used on top of "pandas", we can do some preprocessing in order to remove stop-words, whitespaces, brackets and punctuations and then lowercase all the letters in the strings. Validating the format of email addresses, phone numbers and URLs is one of the most important tasks that is carried out in the cleaner module:

- **Emails:**

Email addresses must comply with some certain rules; each email address should consist of a local and a domain part [11]: "local-part@domain". Therefore, a restrictive filter can be put in place in a way that only email addresses that fall in with the valid email address pattern can be considered. Once again, we can use the Python "re" library to define valid regular expression patterns for emails, in order to filter out (remove) the content that does not follow these rules.

- **Phone numbers:**

Although the format of valid phone numbers may be different for each country, the International Telecommunication Union (ITU) provided some recommendations for defining the format and the length of phone numbers. Also benefiting from the Python library called "phonenumber" which is supported by Google, we can use the "parse" method, which takes two input arguments as a string, one representing the phone number (in any format such as E164, national, international, etc.), while the other represents the country. It neglects all the punctuations and white-spaces and removes all the non-number bits and returns a phone number object. By using the "format_number" method on this phone number object, we can produce the phone numbers in our desired format. We have selected "E.164" which is an international standard and ITU-T Recommendation. According to this standard, the phone number must contain only digits split as: 1 to 3 digits representing the country code followed by subscriber number which is limited to maximum 12 digits, for example: +00 111 222 33 44. Meaning 00 being the country code and 1112223344 being the subscriber number. There are also other features we can use from this

library, such as getting extra information about the location associated with phone numbers which will come handy when double checking the localities assigned to prospects.

- **URLs:**

A reference to a web resource in technical terms, is called a Uniform Resource Locator (URL), it also specifies the location of the resource on a computer network. A URL is a particular type of Uniform Resource Identifier (URI). According to World Wide Web Consortium or W3C (2009), URLs are generally being used to reference pages (http), file transfer (ftp), email (mailto), access to database (JDBC) and so on. The standard format of a URI, which would also be the case for URLs by definition, is given below [14]:

URI = scheme ":" ["//" authority] path ["?" query] ["#" fragment]

- Scheme: protocol used to reach the resource on the world wide web, such as http.
- Host name: also called the domain name of the host where the resource is located.
- Port number: since servers offer more than one service, specification of the required service is indicated by the port number.
- Path: which pinpoints the specific resource inside the host required by the user.
- Query string: the parameters of the search that are needed to be given to the server-side scripts.
- Fragment identifier: which specifies a particular location on a web page.

It is also important to consider the fact that the scheme and the host name parts are not case-sensitive, however the path and query string are case-sensitive. It is common practice to specify the whole URL in lower case.

For example: `https://www.example.com/users?id=1#line=10`

According to all the information we can see above, establishing an admissible pattern for the URLs can be done once more, using the regular expressions that filter out the non-valid formats of URLs. Finally, considering the fact that prospects possessing no valid contact information such as emails, phone numbers, and company URLs, present no possible way of identification, making it difficult to decide whether they are unique or not. This will cause an issue with regards to having duplicated prospects and will later introduce a large bias in analysis of the prospects. There-

fore, another responsibility that falls upon the cleaner module is, list and categorize all the contact information of each prospect, and drop the records that have no values in these lists.

3.2. Taxonomy mapping (Profiler)

Due to having synonyms in almost every language, addressing different businesses offering services might raise various complications. Take the term "electrician", a person who installs and maintains electrical equipment, as an example. One can interpret terms such as lineman, electrical technician, electrical expert, etc., with the same concept and meaning.

The vast diversity of how we can address an occupation, motivates the idea of fabricating organized dictionaries with a mapping logic to relate all similar terms to one specific occupation. This type of process of classifying and categorizing is commonly referred to as "taxonomy".

Based on the needs and capacity of each department of ProntoPro, different taxonomies have been established.

We can outline these taxonomies through a practical example:

A potential client living in the city of Milan is trying to find someone who has the ability and skills to fix the malfunctioning water heating system of their house. Imagine the case that they would do so, by searching a solution to their problem on the Internet, it is going to be more likely for them to address their needs more specifically; in other words, they would not search for the word "plumbers" on the search engines, instead they would insert a statement similar to "repairing water heating system in Milan". In the taxonomy of ProntoPro SEO department, the descriptions used to reference intended landing pages related to these queries are called "Business Class". Business classes are usually obtained from consulting with Google as they are the most commonly used search engine provider across the globe according to statista (86.64 percent in September 2021).

In order to reference detailed and specific tasks and activities done in a certain line of work, the concept of "Tag" is created. As examples, both "water heater installation or replacement" and "Unblocking drains" are tags. A group of coherent tags construct another concept called "Service", and related services are grouped together and form the concept of "Service Groups", also known as "Professions". Finally, the related professions are classified in another entity called "Category".

To clarify the definition, as well as the relation between these concepts and taxonomies, consider the following example:

A customer searches for "fixing water boiler Milano" in Google, this query will result

to land on a page that has been created and mapped to the certain business class of "Heat pump installation" which in turn is related to the tag "water heater installation or replacement"; This tag is mapped to the service called "Plumbing and heating" which belong to the service group "plumbers" in the category of "home improvement".

All these procedures and organizational designs are making it possible to link a request to a merchant possessing necessary skills and adequacy to completely solve the problem of clients.

Figure 3.1, illustrates the relationship between taxonomy of professions in ProntoPro:

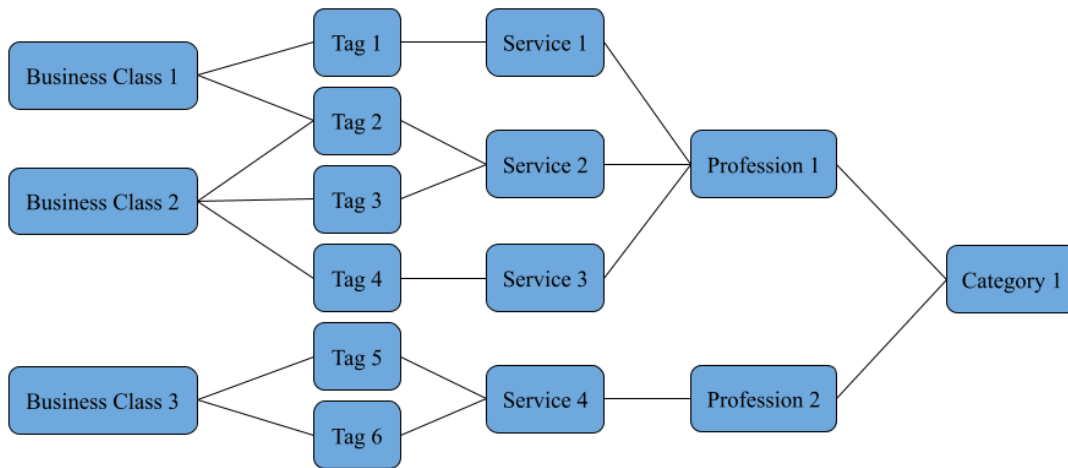


Figure 3.1: Profession taxonomy

Similar to professions, localities (or cities) are also presented in a particular taxonomy. Generally, countries are divided into regions, and regions consist of provinces. In each province, there are multiple cities.

For taxonomy of professions, there exists corresponding tables of business class, tag, service, profession, and category; while for taxonomy of localities, tables of locality, province, and region have been established. The taxonomy tables of each country are separated due to language differences. After the extracted data is cleaned and ready, different procedures must be carried out in order to map the values to the corresponding records in ProntoPro taxonomy and modify them accordingly. From now on we will reference this mapping procedure as "profiling".

The task of profiling is divided into two separate tasks. One being the profiling of the locality fields and the other, profiling of the profession fields.

In order to profile localities and professions of the extracted data, we must consider whether there exists a taxonomy for each of the records, according to their country. Therefore, it is necessary to include certain fields which are required in the profiling pro-

cess. These necessary fields are: country code, locality, profession or keywords (used for profiling professions).

3.2.1. Profiling localities

Certain cities might be known and called by different names. This can happen due to language differences, such as the city of Milan which is called Milano in Italian language, or other factors, such as the case for the city Palma which is also known as Palma de Mallorca.

It can be expected that in the locality taxonomy of ProntoPro, cities are known by the official names given to them by national governments. However, it might be the case that the sources we have crawled and scraped the data from, use different names to reference localities.

Not profiling the localities to the ProntoPro taxonomy, eliminates our chance of having accurate and organized information to be used for analytical and marketing purposes.

One of the things that can be of aid to undergo this task is the zip code. Also known as postal code, a zip code is a series of numbers communicating information about people within different geographic groupings. Looking at official governmental sources, zip code libraries can be found that include useful attributes such as city name, coordinates, province and region names for each zip code. By creating aggregated lookup tables that are formed from the mapping between these zip code libraries and ProntoPro locality taxonomy, we can expect faster and more reliable results to find and profile localities given in our scraped data.

We have considered three different possibilities that a city can be called in these aggregated lookup tables to increase the rate of successfully profiled records.

To compare two string values in Python, equality (`==`) and comparison (`<`, `>`, `!=`, `<=`, `>=`) operators can be utilized. But we have to consider that capital letters “A” and lowercase “a” are not equal according to the equality operator, in other words, they are case-sensitive. Thus, we have to compare the two strings after some modifications have been applied to them; On the other hand, some characters in a string value do not provide any useful information and are often considered as noise, such as brackets ([,]).

By using a Python application called "Python-slugify" on string values, we can transform them into strings that are trimmed, lowercased, etc. Also, white-spaces are going to be replaced by hyphens (-). In order to have higher time efficiency, slugifying the three versions of the localities has been included in the process of creating the aggregated lookup tables.

A different method of comparing two string values in Python can be done by using a class

called "SequenceMatcher" from the "difflib" library. This module provides various classes and functions designed for comparing sequences. A method in the "SequenceMatcher" class is the "ratio", which returns a measure of the sequences' similarity as a float number between the range of 0 to 1. This value is computed from the following formula (3.1):

$$Ratio = 2 \times \frac{M}{T} \quad (3.1)$$

Where T is the total number of elements in both sequences, and M is the number of matches. This value is going to be 1 if sequences are identical and 0 if they have nothing in common. Based on empirical evidence and many trials, an acceptable ratio for considering two strings similar to each other is between 0.80 to 0.85.

The process of profiling localities will iterate through each record of the scraped data, comparing their locality field with the records in the aggregated lookup tables.

Algorithm 3.1 describes the operations and procedures of the process.

The algorithm checks whether a record in the input file (scraped and cleaned data) having the same locality has already been profiled by the algorithm. If so, we propagate the same profiling results to the current record and move on to the next iteration. This step will increase the speed of the algorithm drastically, due to the fact that scraped data is commonly extracted in batches that have the same localities. Also by using zip codes, we are limiting the number of comparisons needed to find a match.

Algorithm 3.1 Profiling Localities

i : input,*lookupSet* : all three version of slugified localities of the lookup tables,*profiledSet* : profiled records of the input file,*propagate(record)* : profile the record according to the already profiled ones with the same locality,*similar(record)* : rows from lookup tables with the same zip code as record.zip,*profile(record, set, operators)* : compare record.locality with all elements of a given set using the specified operators and profile the record accordingly. returns True in case of success.

```

1: for record ∈ i do
2:   if record ∈ profiledSet then
3:     propagate(record)
4:     record → profiledSet
5:     continue
6:   else
7:     slugify(record.locality)
8:     sameZipSet = similar(record)
9:     if sameZipSet ≠ ∅ then
10:      if profile(record, sameZipSet, (equality, SequenceMatcherratio)) then
11:        record → p
12:        continue
13:      else
14:        Go to 14
15:      end if
16:    else
17:      if profile(record, lookupSet, (equality, SequenceMatcherratio)) then
18:        record → p
19:        continue
20:      else
21:        Flag the record as un-profitable.
22:      end if
23:    end if
24:  end if
25: end for

```

3.2.2. Profiling professions

Similar to the process of profiling locality fields, specific aggregated lookup tables are essential for profiling the professions.

Different inter-related taxonomies across departments of ProntoPro that are related to professions, can be used to generate the proper aggregated lookup tables. Contemplating on the structure and relations between these taxonomies, led us to design these lookup tables with the logic of taking business classes, services, and tags as keywords related to their correlated profession. Once again, to speed up the process, we store the slugified form of both the professions and keywords in the aggregated lookup tables.

The scraped data must contain some content regarding the potential occupation of the prospects, these contents will usually be store in a column called keywords. Although, it might be observed, that some of the sources used for scraping data, almost use similar taxonomy to reference a line of work. We can take advantage of this opportunity to speed up the process of profiling professions by storing them in a column called profession which will have a higher priority to be compared to the actual professions in ProntoPro taxonomy.

Same as profiling localities, the process of profiling professions will iterate through each record of the scraped data, comparing their keywords or profession fields with the records in the aggregated lookup tables.

In algorithm 3.2, you can find details about the process of profiling the profession fields. Once again, propagating the profiling results of an already profiled record in the same input file leads to speeding up the process.

Algorithm 3.2 Profiling Professions

i : input,
professionSet : slugified professions of the lookup tables,
keywordSet : slugified keywords of the lookup tables,
profiledSet : profiled records of the input file,
propagate(record) : profile the record according to the already profiled records with the same keywords and profession,
profile(record, set, operators) : compare record.profession with all elements of a given set using the specified operators and profile the record accordingly. returns True in case of success,
elect(record, set, operators) : compare all record.keywords with all elements of a given set using the specified operators and profiles the record if the most frequent profession associated with the matched keywords has a higher frequency than the existing profileRanking otherwise profile the record with its highest profileRanking

- 1: **for** *record* \in *i* **do**
- 2: **if** *record* \in *profiledSet* **then**
- 3: *propagate(record)*
 record \rightarrow *profiledSet*
 continue
- 4: **else**
- 5: *slugify(record.profession)*
- 6: **if** *profile(record, professionSet, equality)* **then**
- 7: *record.profileRanking* = 10
- 8: **else**
- 9: **if** *profile(record, professionSet, SequenceMatcherratio)* **then**
- 10: *record.profileRanking* = 5
- 11: **else**
- 12: Go to 16
- 13: **end if**
- 14: **end if**
- 15: *slugify(record.keywords)* \rightarrow *keywords*
- 16: *elected* = *elect(record, keywordSet, (equality, SequenceMatcherratio))*
- 17: **if** *elected* **then**
- 18: *record* \rightarrow *profiledSet*
 continue
- 19: **else**
- 20: Flag the record as un-profilable.
- 21: **end if**
- 22: **end if**
- 23: **end for**

3.3. Data Validation (Data pipeline)

As already mentioned, having prospects that contain no contact information, limits our ability to identify them uniquely which will reduce the accuracy of analysis on them and eliminates the possibility to reach out to all prospects stored in our database. This problem is dealt with in the cleaner module in section 3.1; however, validity of contact information still remains undetected.

Contact information such as phone numbers and email addresses, are invalid if they do not exist in their specified domains.

By establishing a series of modules for filling out missing contact information and validating them using proper tools, we can make sure more reliable information is being collected and uploaded to the database. The name "Data pipeline" has been given to these series of modules which are briefly described in the following:

3.3.1. Company URL checker

Using an elegant and simple HTTP library for Python called "requests", one can easily send HTTP requests. Each scraped, cleaned, and profiled record might have a value in their company URL field. By passing this URL as an argument to the "get" method of the "requests" library, a response object is returned. From this object, we can access the status code of the response. If this value is equal to 200, it means that the request was successful, that can be interpreted as the URL being valid to us.

3.3.2. Email Debouncer

An integration to an email validation service called debounce.io has been implemented in this module, which offers straightforward API access for various programming languages including Python. For each email address of each record of prospects, an API call can be made which returns a response object of the "requests" library. Using the "load" method from the "json" library in order to convert the response into a Python dictionary, we can access the result code which indicates the status of the inserted email address. In case of it being equal to 5 (deliverable), we consider the email as valid.

Since this service is not provided for free and might take a long time, it is necessary to store the results of each API call with its corresponding email address in a database table called email lookup dictionary. This will reduce the overall cost of email address validation. To elaborate the reason behind this choice, imagine the same prospect with the same email address has been extracted from two different sources, one API call to check the validity of this prospect's email address is sufficient.

In conclusion, for each scraped, cleaned, and profiled record, we initially check the email lookup dictionary, if the email address exists in this file and is indicated as valid (or deliverable), it is considered as valid. In case it is specified as invalid, if the latest validity check is more than 1 year old, we make another API call and update the record in the email lookup dictionary; if not, we consider the email as invalid.

3.3.3. Email scraper

There is a high chance of finding email addresses on a company website. For prospects that have company URLs but are missing email addresses, scrapping might be a good solution. Process starts by searching through the company websites trying to find email addresses using regular expressions, then validating the scraped emails by searching them in the email lookup dictionary or using the email debounce API on them before updating the email lookup dictionary. To be practical, the process of scraping for emails will have a time limit of 5 seconds.

3.3.4. Email guesser

Based on other information available for each prospect that is missing email addresses, generating email addresses might be a good option. However, this process is utterly challenging and only a prototype of the supposedly functional module is implemented in a way that only for prospects containing company URLs and no email address, company domain is extracted and later used to generate an email address like info@domain. The validity of the generated email address is checked by using the email lookup dictionary or the email debounce API.

4 | Using the extracted data and application features

In this chapter, you can find how the extracted data is being utilized. Also various features of the application is introduced.

4.1. Prospects market overview

By having the cleaned and profiled records collected in the database, it is possible to design insightful dashboards and graphs to visualize this data. These graphical demonstrations can be used to make better and more accurate business decisions.

Instead of implementing custom tools for having these graphical visualizations, we decided to benefit from ProntoPro' s subscription at Tableau.

Tableau is a visual analytics platform transforming the way we use data to solve problems, empowering people and organizations to make the most of their data. As the market-leading choice for modern business intelligence, their analytics platform makes it easier for people to explore and manage data, and faster to discover and share insights that can change businesses and the world [21].

In order to avoid having misleading information in our visualizations, it is required to take some necessary actions; One is being the process of flagging the prospects status. This process is done by designing proper SQL queries to flag the prospects according to their contact tables (email, phone, company_url); The importance of this process is the result of being limited to not having a live connection between our database and the Tableau data source since the data is too large which makes the process costly. It is also important to avoid pushing private contact information of prospects to the Tableau. The data is pushed to the Tableau once a week.

Another useful feature of the Tableau is being able to design customizable filters which will later give the clients the possibility to manage and manipulate the graphical visualizations according to their needs. In Figure 4.1, you can see an example of the prospects market overview. Also, in Figure 4.2, you can see the distribution of the prospects over the map

categorized by their status. Clients can easily navigate through the profession or locality of their interests to obtain the visualizations they are looking for.

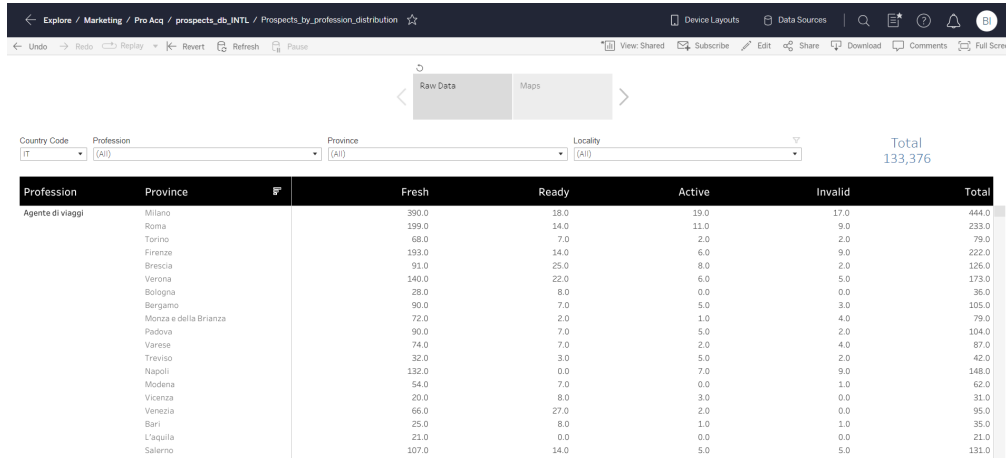


Figure 4.1: Prospects market overview

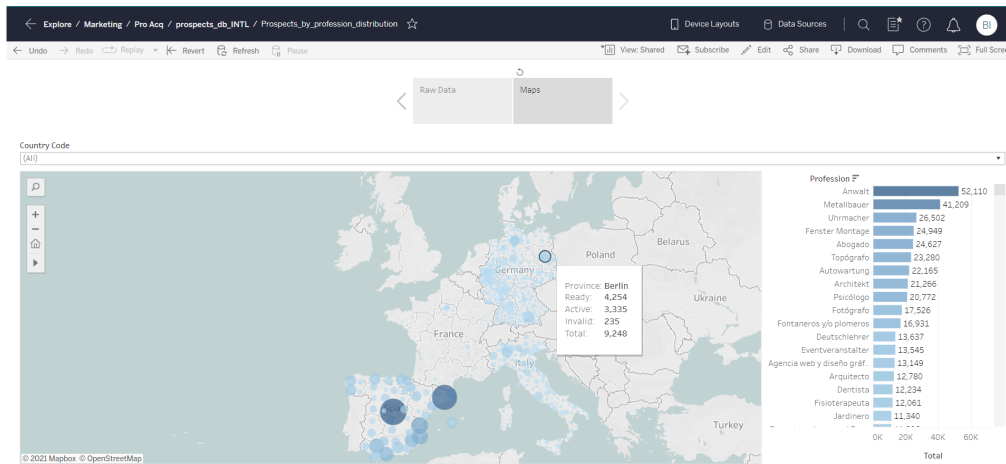


Figure 4.2: Prospects market overview map

4.2. Count and extract modules

The data stored in the database must be provided to the ProAcquisition team which has the responsibility to contact and attract potential merchants through different channels such as email addresses or phone numbers. Each prospect might have multiple contact information with different statuses (ready, active, invalid, subscribed). The ProAcquisition team is interested in extracting prospects via their contact points flagged as ready that will be later used in their marketing campaigns. This is also expected that they need a

tool to have an overall view on how many contact points are available before they decide on how many of which profession or locality to use. The "count" module is implemented to provide them with this insight; While the "extract" module is responsible for both retrieving and setting the extracted contact point statuses as active since they presumably are going to be used as soon as they are extracted.

The following example will help to understand the process better:

Client uses the "count" feature with arguments `contact=email`, `status=ready`, `country=IT`, and `profession_id=1`. The result is the number of ready prospects reachable through an email address residing in Italy that are assigned with `profession_id` equal to 1 which is "Organizzazione eventi e feste" or "Organization of events and parties".

Based on the number of available prospects which satisfy these requirements, clients can extract the desired amounts of prospects which automatically results in changing their email statuses to "active". Keep in mind that changing the status of a contact belonging to a prospect does not necessarily change the overall status of that prospect. In ProntoPro website, each merchant sign-up page is designed specifically for every profession and localities, thus when extracting prospects from the database, it would be incredibly helpful to provide URLs to access the relevant sign-up pages according to the prospects profession and locality. The corresponding sign-up URLs are created by the "extract" module. This feature is proven to be very useful when extracting prospects for digital email marketing or DEM, the sign-up page URLs will be provided to our potential customers which makes the process of the signing up easier. The results of the extract module will be stored in files with .csv format. Also, an integration with a CRM (Customer Relationship Management) tool called Customer.io is accomplished in which the process starts by the user inserting a .csv file containing all the filter conditions and information needed to execute the "extract" module. Iterating through all the inserted queries and pushing the results of each extraction to the CRM tool using the provided APIs, improves both the time efficiency and the reliability of the process.

4.3. Synchronization with other integrations

As previously discussed, contact points of prospects can belong to one of the following classes:

1. **Ready**: when the prospect is newly uploaded to the database via a specific contact point.
2. **Active**: when the prospect is extracted via a specific contact point.
3. **Invalid**: when the contact point cannot be used anymore.

4. **Subscribed:** when the prospect associated with the contact point has subscribed to ProntoPro.

There exist various reasons that are sufficient to consider a contact point as invalid. One of which is the blacklisted. A contact point is blacklisted if the person who has the ownership of the contact information does not give their consent to us keeping their contact or personal information. To solve this problem, their contact information is irreversibly encrypted and kept in a table in the database called blacklist. Whenever a prospect is being uploaded, their encrypted contact information will be compared to the content of this table. The prospects with blacklisted contact points will be discarded.

Moreover, a prospect must be flagged as invalid, if they turned down the offer to join ProntoPro after being contacted. Also, when merchants unsubscribe from ProntoPro platform, they should not be contacted again, thus must be considered as invalid. A contact point is considered to be "subscribed" when the prospect associated to it has signed-up to ProntoPro and is known as a merchant now. In this case, all the other contact point associated to this prospect should be flagged as "subscribed" and must not appear in extract module results.

Another important scenario takes place when a prospect has been extracted and its contact point has been flagged as active for more than 30 days, but no response has been received from them yet. This contact can potentially be flagged as ready in order to be extracted and contacted again. However, the maximum number of times a contact can change from active to ready must be limited, the threshold decided for this restriction is equal to five. Figure 4.3 represents the status life cycle of a contact point.

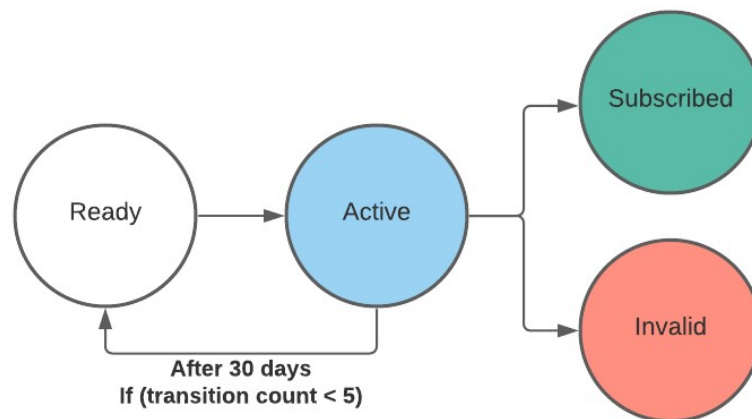


Figure 4.3: Contact status life cycle

All the procedures that are necessarily required to change the status of contact points or overall status of the prospects, are implemented in the "sync" module. It can be

anticipated that multiple continuous integrations with other departments of ProntoPro must be done in order to achieve the goal of consistency and reliability. The task of updating dictionaries regarding the taxonomy profiling has been included in the "sync" module as well. The "sync" module is scheduled to be executed daily, every 4 hours. It is being hosted on an internal server machine at ProntoPro.

4.4. User interface

Having an intuitive user interface is essential to increase the usability of almost all software applications of any sort. By using the Python toolkit called "tkinter" to implement a graphical user interface (GUI), we have simplified the usage of the application. The colors and fonts used to shape the GUI has been provided by the design team of ProntoPro. In Figures 4.4 and 4.5, you can see the final results for the main and the extract menu respectively.

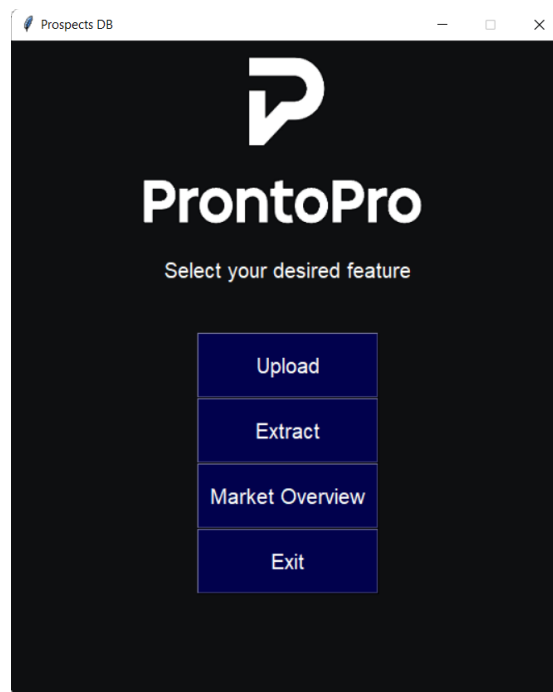
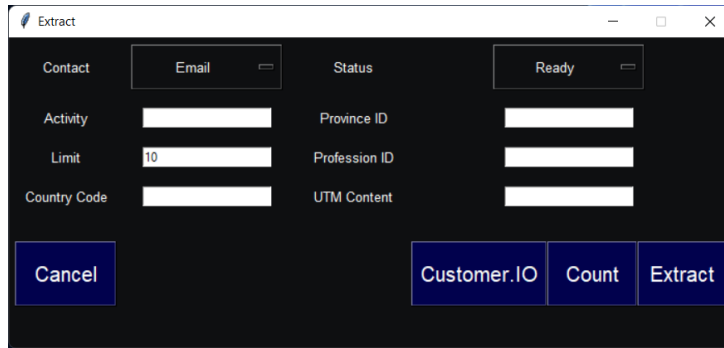


Figure 4.4: Main menu



The image shows a software window titled "Extract" with a dark background. It contains several input fields and buttons. The fields are arranged in a grid-like structure:

Contact	Email	Status	Ready
Activity		Province ID	
Limit	10	Profession ID	
Country Code		UTM Content	

At the bottom, there are four buttons: "Cancel", "Customer.IO", "Count", and "Extract".

Figure 4.5: Extract menu

5 | Conclusion and future work

The final result of this project appeared to be as useful as expected both for high level decision making as well as using the acquired data in order to communicate properly with the prospects. There are more than 5,546,972 prospects in the database having 1,928,418 distinct emails, 5,841,910 phone numbers, and 2,570,226 company URLs in their respective tables.

However, certain measures needed to be taken in order to increase the usability of the application such as improving the time efficiency of all the modules.

For processes that include the task of iterating through a dataframe, using the "itertuples" method of the "pandas" was a suitable choice; You can find the result of some benchmarks in Figure 5.1 [13]. Concurrency can also be used to improve the performance. We used "concurrent.futures" for this purpose.

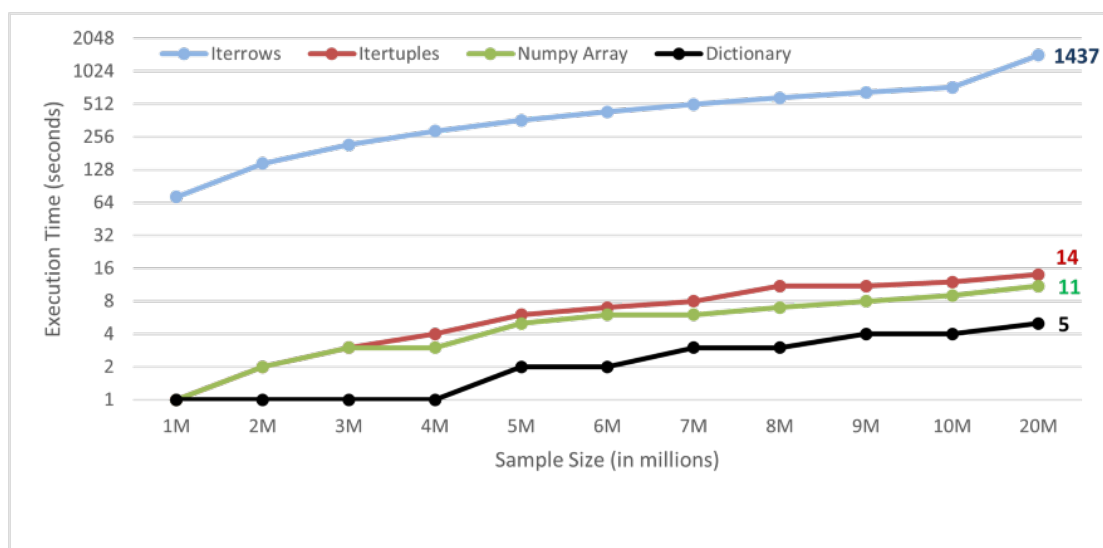


Figure 5.1: Benchmark iteration time

Also, in order to decrease the query time of the database, pre-calculated overall prospect statuses seemed to be helpful for some cases such as prospect market overview.

There seems to be a great opportunity to improve the profiling logic by using artificial

intelligence techniques such as NLP (Natural Language Processing) in order to map the external taxonomies to the internal ones. Additionally, implementing a decision support system (DSS) using recommender systems can make it possible to make better informed decisions in a shorter time.

Bibliography

- [1] Alan Ross Machinery Corporation v. Machinio Corp., 2018.
- [2] Associated Press v. Meltwater US Holdings, Inc., 2013.
- [3] Craigslist, Inc. v. 3Taps Inc., 2013.
- [4] diffbot, 2019. URL <http://www.diffbot.com/>.
- [5] A. J. Dreyer and J. Stockton. Internet “data scraping”: A primer for counseling clients. *New York Law Journal*, pages 1–3, 2013.
- [6] Facebook, Inc. v. Power Ventures, Inc., 2016.
- [7] J. I. Fernández-Villamor, J. Blasco-García, C. A. Iglesias, and M. Garijo. A semantic scraping model for web resources-applying linked data to web page screen scraping. In *International Conference on Agents and Artificial Intelligence*, volume 2, pages 451–456. SciTePress, 2011.
- [8] D. Glez-Peña, A. Lourenço, H. López-Fernández, M. Reboiro-Jato, and F. Fdez-Riverola. Web scraping technologies in an api world. *Briefings in bioinformatics*, 15(5):788–797, 2014.
- [9] hiQ Labs, Inc. v. LinkedIn Corp., 2019.
- [10] Intel Corp. v. Hamidi, 2003.
- [11] J. Klensin et al. "simple mail transfer protocol", 2008.
- [12] V. Krotov and L. Silva. Legality and ethics of web scraping. *Communications of the Association for Information Systems*, 2018.
- [13] S. Kumar. Here’s the most efficient way to iterate through your pandas dataframe, 2021. URL <https://towardsdatascience.com/search?q=Here%E2%80%99s%20the%20most%20efficient%20way%20to%20iterate%20through%20your%20Pandas%20Dataframe>.

- [14] R. T. F. Larry Masinter, Tim Berners-Lee. Uniform resource identifier (uri): Generic syntax, 2005.
- [15] W. Mckinney. pandas: a foundational python library for data analysis and statistics. *Python High Performance Science Computer*, 01 2011.
- [16] J. Mingers and G. Walsham. Toward ethical information systems: The contribution of discourse ethics. *MIS quarterly*, pages 833–854, 2010.
- [17] S. Munzert, C. Rubba, P. Meißner, and D. Nyhuis. *Automated data collection with R: A practical guide to web scraping and text mining*. John Wiley & Sons, 2014.
- [18] D. Possler, S. Bruns, and J. Niemann-Lenz. Data is the new oil—but how do we drill it? pathways to access and acquire large data sets in communication science. *International Journal of Communication (19328036)*, 13, 2019.
- [19] A. Sellars. Twenty years of web scraping and the computer fraud and abuse act. *BUJ Sci. & Tech. L.*, 24:372, 2018.
- [20] V. Singrodia, A. Mitra, and S. Paul. A review on web scrapping and its applications. In *2019 International Conference on Computer Communication and Informatics (IC-CCI)*, pages 1–6. IEEE, 2019.
- [21] Tableau, 2003. URL <https://www.tableau.com/why-tableau/what-is-tableau/>.
- [22] the Unicode Consortium; edited by the Unicode Consortium. — Version 14.0, 2021.
- [23] Ticketmaster LLC v. PRESTIGE ENTERTAINMENT, 2018.
- [24] M. L. Zachary Gold. Robots welcome? ethical and legal considerations for web crawling and scraping. *Wash. JL Tech. & Arts*, 2018.
- [25] Z. Zhou and M. Mashuq. Web content extraction through machine learning. *Stanford University*, pages 1–5, 2014.
- [26] S. Zuboff. Big other: Surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology*, 30(1):75–89, 2015.

A | Appendix A

Here you can find all the services offered by ProntoPro:

'Lawyer', 'Building manager', 'Travel agency', 'Marketing and advertising agency', 'Webagency and graphic design', 'Mason', 'Rental of premises and equipment for events', 'Event entertainer', 'Antenna technician', 'Architect', 'Craftman', 'Insurer', 'Financial Advisor', 'Personal Assistant', 'Driving School', 'Carpenter', 'Catering', 'Food delivery and cooking teacher', 'Energy certifier', 'Life coach', 'Pool constructor', 'DJ, Musician or Musical Group', 'Dentist', 'Developer and programmer', 'Dietitian and nutritionist', 'Fashion designer and fashion school', 'Electrician', 'Cleaning company', 'Vehicle rental company', 'Construction company', 'Security and surveillance company', 'Endocrino', 'Personal Trainer', 'Shipping and transport', 'Specialist in thermal insulation and humidity', 'Pest control specialist', 'Health Specialist', 'Tourism and hotel specialist', 'Beautician', 'Stylist', 'Art expert', 'Renewable and environmental energy expert', 'Spanish language expert', 'Glass and glazing expert', 'Curtain maker', 'Window and door manufacturer', 'Physiotherapist', 'Plumber', 'Photographer', 'Funeral services', 'Blacksmith and/or Locksmith', 'IT', 'Elevator installer', 'Flooring installer', 'Painter', 'Dance Instructor', 'Yoga and Pilates instructor', 'Martial arts and boxing instructor', 'Aviation instructor', 'Sport coach', 'Swimming coach', 'Interior designer', 'Gardener and Landscaper', 'Vehicle cleaner', 'Massage therapist', 'Mechanic', 'Mover', 'Naturopath', 'Notary', 'Event organizer or agency', 'Drone pilot', 'Chiropodist', 'Video Maker', 'German teacher', 'Singing teacher', 'Chinese teacher', 'Korean teacher', 'French teacher', 'Greek and Latin teacher', 'Guitar teacher', 'Languages teacher', 'Computer science and programming teacher', 'English teacher', 'Musical instrument teacher', 'Japanese teacher', 'Sign language teacher', 'General Music Teacher', 'Norwegian teacher', 'Piano teacher', 'Portuguese teacher', 'Russian teacher', 'Violin teacher', 'Arabic teacher', 'Private teacher and tutorships', 'Psychologist', 'Psychiatrist', 'Chiropractor', 'Sewer repairer', 'Printing services', 'Animal Services', 'Bicycle services', 'Home appliance repairer', 'Upholsterer', 'Tattoo and piercings', 'Dry Cleaning and Laundry', 'Surveyor', 'Translator and interpreter', 'Veterinary', 'Plasterer'

List of Figures

1	High level workflow of the solution	2
1.1	ER diagram	7
3.1	Profession taxonomy	22
4.1	Prospects market overview	32
4.2	Prospects market overview map	32
4.3	Contact status life cycle	34
4.4	Main menu	35
4.5	Extract menu	36
5.1	Benchmark iteration time	37

