INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Praxis Market Drift

MIGUEL MORIM OLIVEIRA
Outubro de 2020

POLITÉCNICO
DO PORTO

Praxis Market Drift

Miguel Oliveira

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Computer Systems

Supervisor: Nuno Filipe Fonseca Vasconcelos Escudeiro

Porto, October 15, 2020

Dedicatory

To my family and friends, whom without their continuous support this dissertation would have not been possible.

Abstract

Over the last decade, digital data has been growing exponentially. Unstructured data is rapidly outgrowing structured data, and so managing unstructured data is increasing as a challenge for many organisations.

Consequently, text mining has been gaining traction as a way to deal with unstructured data. Text mining is a form of data mining that deals with text and is the process of transforming unstructured text into meaningful and actionable information.

Praxis is a platform that implements a virtual market for project/internship offers. organisations submit their project/internship offers that become available for search, and students search the platform using keywords that express their interest. Praxis has loads of unexplored data from which they can extract useful information to obtain more insights on their internships market.

This dissertation proposes a solution based on text mining techniques, that displays the necessary information to analyse the evolution of users' interests and internship offers submitted in Praxis, over time.

Keywords: text mining, unstructured data, preprocessing, word frequency, clustering

Resumo

Durante a última década, a quantidade de dados digitais tem vindo a crescer exponencialmente. A quantidade de dados não estruturados está rapidamente a superar a quantidade de dados estruturados, e portanto, a gerência de dados não estruturados está a crescer como um desafio para várias organizações.

Consequentemente, *text mining* tem vindo a ganhar tração como forma de lidar com dados não estruturados. *Text mining* é uma forma de data mining que trabalha com texto e é o processo the transformar dados não estruturados em informação significativa.

O Praxis é uma plataforma que implementa um mercado virtual para ofertas de projetos/estágios. Organizações submetem as suas propostas de projeto/estágio que ficam disponíveis para pesquisa, e os estudantes pesquisam na plataforma, com recurso a palavras-chave que expressam os seus interesses. O Praxis tem muitos dados inexplorados dos quais eles podem extrair informações úteis para obter uma melhor perceção do seu mercado de estágios académicos.

Esta dissertação propõe uma solução baseada em técnicas de *text mining*, que apresenta a informação necessária para analisar a evolução dos interesses dos utilizadores e das ofertas de estágio no Praxis, ao longo do tempo.

Acknowledgement

I would like to thank everyone that supported and helped me throughout my academic journey.

To my family, especially my parents, for the continuous support along the journey that culminated in this dissertation.

To my supervisor Nuno Escudeiro that guided me through this dissertation, for the support and suggestions that ensured I would produce my best work.

Lastly, to my friends, for the support and motivation along all these years.

Contents

List of Figures

List of Tables

List of Algorithms

List of Source Code

List of Acronyms

AHP	Analytic Hierarchy Process.
API	Application Programming Interface.
CSS	Cascading Style Sheets.
DCS	Document Clustering Server.
FFE	Fuzzy Front End.
GAAC	Group Average Agglomerative Clusterer.
HTTP	Hypertext Transfer Protocol.
IDC	International Data Corporation.
NCD	New Concept Development.
NLP	Natural Language Processing.
POS	Part-Of-Speech.
REST	Representational State Transfer.
STC	Suffix Tree Clustering.
SVD	Singular Value Decomposition.
SVG	Scalable Vector Graphics.
UML	Unified Modeling Language.
VSM	Vector Space Model.
WSGI	Web Server Gateway Interface.
YAML	YAML Ain't Markup Language.

Chapter 1

Introduction

1.1 Context

Over the last decade, digital data has been growing exponentially. The Global DataSphere quantifies and analyses the amount of data created, captured, and replicated in any given year across the world (IDC 2020). According to a study conducted by the International Data Corporation (IDC), the Global Datasphere is expected to grow from 45 Zettabytes (1 ZB = 10^{12} GB) in 2019 to 175 Zettabytes by 2025 (Reinsel, Gantz, and Rydning 2020). The same study also suggests that more than 80% of the total data will be unstructured data.

Managing unstructured data is growing as a challenge. These statistics can be alarming for enterprise organisations that are already struggling with the issue of having to deal with loads of unstructured data.

Structured data is the type of data that adheres to a pre-defined data model and is therefore straightforward to analyse (Enterprise Big Data Framework 2019). Names, dates and addresses are examples of structured data. It is the type of data found on relational databases and spreadsheets, highly organised and easily understood by machine language. Unstructured data is the type of data that either does not have a predefined data model or is not organised in a pre-defined manner (Enterprise Big Data Framework 2019), therefore it is difficult to understand using traditional tools and methods as compared to structured data. Text documents, video and audio are examples of unstructured data.

Since most of the data is unstructured, it is advantageous for an organisation to turn such data into meaningful information. By using the information and knowledge extracted from unstructured data, organisations are able to get deeper insights on their business, that can be leveraged for informed decision making and improving customer relationships.

However, finding this previously undiscovered information buried within unstructured data is not an easy task. It requires advanced analytics and a high level of technical expertise.

This process of transforming unstructured data into structured data through the identification and exploration of large amounts of text, so the data that can be analysed in a traditional way is known as text mining (Expert System 2016). Text mining applies many

techniques to extract useful information and knowledge hidden in text content, revealing patterns, trends and insights in large amounts of information.

1.2 Problem

Praxis (Praxis Network 2020) is a European platform that implements a virtual market for project/internship offers and is run by the Praxis network, a consortium of higher education institutions, companies, associations, research labs and chambers of commerce. These institutions submit their project/internship offers that become available for search. Students search the platform using keywords that express their interest.

In recent years, the number of students enrolled in universities and higher-education institutions outside their countries of citizenship has risen dramatically. Over 4.5 million students were enrolled in higher education outside their home countries in 2015, five times more than the 0.8 million in 1975 (Relocate Magazine 2019).

These statistics show that platforms like Praxis are expected to gain more and more value as international student mobility rates keep growing.

In order to offer the best possible experience to their users, both students and companies, it is extremely important that Praxis is able to understand the evolution of their own internships market. For example, knowing that a large influx of students are actively looking for internship offers in a certain area of study, Praxis is able to reach out to companies in that area of study that might be interested in submitting offers, with the potential of acquiring a new partner for their network.

Given the influx of users, it is expected that Praxis has loads of unexplored data from which they can extract useful information to obtain more insights on their business. Therefore, Praxis needs a way to extract this useful information so they can analyse and understand the internships market and its evolution over time.

1.3 Objectives

This thesis has the main objective of developing a solution directed at Praxis administrators, named Praxis Market Drift, capable of displaying the necessary information to analyse the evolution of users' interests and internship subjects submitted, over time.

To accomplish this goal, a study must first be conducted in the domain of text mining, in order to understand how text mining can be applied in the context of this thesis and which techniques should be used to process the data. The study should also explore the various types of data visualisation available, so the information is displayed in the most effective way. The result of this study should then be reflected on the implementation of a solution

capable of displaying representative views of the internships market evolution for posterior analysis, based on text mining techniques.

1.4 Contributions and expected results

This dissertation aims to demonstrate that the application of text mining techniques can help businesses derive high-quality information from unstructured text data, and so the expected result is a flexible tool that uses text mining techniques to provide information about Praxis internships market and its evolution over time.

1.5 Value analysis

As already mentioned, managing unstructured data is growing as a challenge, and extracting information from unstructured data gives many benefits and advantages to organisations. organisations are able to get deeper insights on their business, that can be used to improve the decision making.

This dissertation has the main purpose of using text mining techniques to extract meaningful information about the Praxis internships market. So, it is safe to assume this work has business value. The Praxis administrators can use this system to analyse the evolution of their market, which in turn improves the decision making, providing the users of their platform a better experience.

This topic will be covered in more detail in Chapter 2.

1.6 Document structure

The first chapter provides an introduction to this dissertation, presenting its context, the problem at hand and the objectives it aims to achieve. Additionally, it presents the contributions and expected results, a brief overview of the value analysis of the project and the structure of this document.

The second chapter presents the value analysis of the solution, where the main goal is to figure out if this project demonstrates the potential of creating value.

The third chapter presents the state of the art related to the domain of text mining. This chapter addresses various themes and concepts associated with text mining, such as text mining techniques, data visualisation as a complement to text mining techniques and technologies that can be used to develop text mining solutions. Lastly, it presents existing solutions that present features related to the objectives of this thesis.

The fourth chapter describes the analysis and design of the solution, and it starts by defining the functional requirements for the system, followed by an analysis of the data available in

the Praxis database. It proposes two different architectures for the system and describes the selection process used to select the architecture. The chapter culminates with a high-level view of the various processes in the system.

The fifth chapter describes all the steps taken and all the technologies used to implement each one of the components defined in the selected architecture.

The sixth chapter evaluates the solution with a satisfaction survey and evaluates the clustering algorithms used in this thesis.

The seventh chapter presents the conclusions of this dissertation, namely the achieved objectives, the limitations faced during the development of the solution and suggestions for future work.

Chapter 2

Value analysis

Value analysis is a systematic and structured approach for improving projects, products and processes, that helps achieve an optimum balance between function, performance, quality, safety and cost. The proper balance results in the maximum value for the project (SAVE 2020).

This chapter has the objective of analysing and using techniques to figure out if this project demonstrates the potential of creating value.

2.1 New Concept Development (NCD) model

The NCD model provides a common language and definition of the key components of the Fuzzy Front End (FFE) (Koen et al. 2002). The FFE is defined as the period between when an opportunity for a new product is first considered, and when the product idea is judged ready to enter "formal" development (Florén and Frishammar 2010).

According to Koen et al. (2002), the NCD model defines five key elements:

- **Opportunity Identification**.
- **Opportunity Analysis**.
- **Idea Generation and Enrichment**.
- **Idea Selection**.
- **Concept definition**.

The following sections describe each one of the key elements according to this project's theme.

2.1.1 Opportunity Identification

Approximately 80% of the total existing data worldwide is in an unstructured format. As the amount of data continues to grow, the amount of unstructured data also continues to grow, and so managing unstructured data is progressively growing as a challenge for many

organisations. By extracting information from within the unstructured data, organisations are able to get deeper insights on their business, thus gaining competitive advantages over organisations that do not manage their unstructured data.

Several methods/techniques can be utilised to assess this key element (Koen et al. 2002):

- Roadmapping - capture the driving forces of the business in graphical form in order to enhance communication and insight.
- Competitive intelligence analysis - practice of collecting, analysing, and communicating the best available information on competitive trends.
- Market research.
- Scenario planning - provides a disciplined approach for imagining and preparing for the future.

2.1.2 Opportunity Analysis

The global data mining tools market is growing predominantly as the amount of data every second is being drastically elevated globally and is forecast to reach USD 1,431.5 Million by 2026 (Reports and Data 2020). This means that more and more organisations are starting to realise the importance of the valuable insights they can get from using data mining tools.

The methods/techniques used for the identification of opportunities can also be used for this key element, but the effort needs to be expanded in considerably more detail. In opportunity identification, these methods/techniques are used to determine if an opportunity exists. In this element, considerably more resources are expended, providing more detail on the appropriateness and attractiveness of the selected opportunity (Koen et al. 2002).

2.1.3 Idea Generation and Enrichment

Praxis is a platform that handles a virtual market of academic project/internship offers. Plenty of projects/internships are submitted and searched for everyday. Naturally, this means that Praxis is most likely currently holding a lot of unexplored data, from which they can extract useful information about their market. This problem incentivised the development of a solution capable of extracting this information from the unexplored data.

This key element can be assessed with the following methods/techniques (Koen et al. 2002):

- Market and business needs and issues continuously interspersing with the technology advances.
- Identifying new technology solutions.
- A variety of incentives to stimulate ideas.

- A formal role for someone (i.e., process owner) to coordinate ideas from generation through assessment.
- A limited number of simple, measurable goals (or metrics) to track idea generation and enrichment.
- Inclusion of people with different cognitive styles on the idea enrichment team.

2.1.4 Idea Selection

After conducting a study on idea generation and enrichment, the next step is selecting which ideas to pursue in order to achieve the most business value. The main goal of this work is the development of a solution based on text mining techniques. The technologies selected for its development are described in Chapter 3.

These methods/techniques can be utilised to assess this key element (Koen et al. 2002):

- Portfolio methodologies based on multiple factors, such as technical success probability, commercial success probability, reward, strategic fit, strategic leverage.
- Formal idea selection process with prompt feedback to the idea submitters.
- Use of options theory to evaluate projects.

2.1.5 Concept definition

The solution to be developed consists of a web application based on text mining techniques, that provides the necessary information to analyse the evolution of users' interests and internship subjects submitted, over time. The Praxis administrators are then able to obtain more insights on their market, that can be used to provide the best possible experience to their users, both students and companies.

This key element can be assessed with the following methods/techniques (Koen et al. 2002):

- Goal deliberation approaches: time spent on carefully defining the project goals and outcomes.
- Setting criteria for the corporation that describe what an attractive project looks like.
- Rapid evaluation of high-potential innovations.
- Early involvement of the customer in real product tests.
- Partner outside of areas of core competence.
- Pursue alternative scientific approaches.

2.2 Value

Value can be defined as any type of good, service, or act that satisfies a need or provides a benefit, which may be tangible or intangible (Haksever, Chaganti, and Cook 2004).

2.2.1 Value for the customer

Value for the customer is any demand-side, personal perception of advantage arising out of a customer's association with an organisation's offering, and can occur as reduction in sacrifice, presence of benefit, the resultant of any weighed combination of sacrifice and benefit, or an aggregation, over time, of any or all of these (Woodall 2003).

2.2.2 Perceived value

Customer perceived value can be defined as the consumer's overall assessment of the utility of a product based on perceptions of what is received and what is given (Li and Green 2011).

Table 2.1 presents the benefits and sacrifices for the customer in the context of this project.

Table 2.1: Benefits and sacrifices for the customer.

Benefits	**Sacrifices**
Process large amounts of data	Hosting servers cost
Analyse the evolution of users' interests	Development cost
Analyse the evolution of internship offers	
Get new insights on the business	
Enhance business decisions	

2.3 Value Proposition

A value proposition is an explicit promise made by a company to its customers that it will deliver a particular bundle of value creating benefits (Hassan 2012).

This project's value proposition consists of a web application based on text mining techniques and directed at Praxis administrators, that provides the necessary information to analyse the evolution of users' interests and internship subjects submitted, over time. The application displays representative views of the internships market evolution for further analysis. The Praxis administrators are then able to obtain more insights on their market, that can be used to provide the best possible experience to their users, both students and companies.

2.4 Business Model Canvas

The business model Canvas is a tool that helps understand a business model in a straightforward, structured way (Amendo 2018). This model is important because it shows how an organisation works, facilitating the definition of ideas and the organisation of actions through a graphic representation of variables, that together, show the value of the business in question.

Figure 2.1 presents the business model Canvas, that proposes a business model according to the theme of this dissertation.

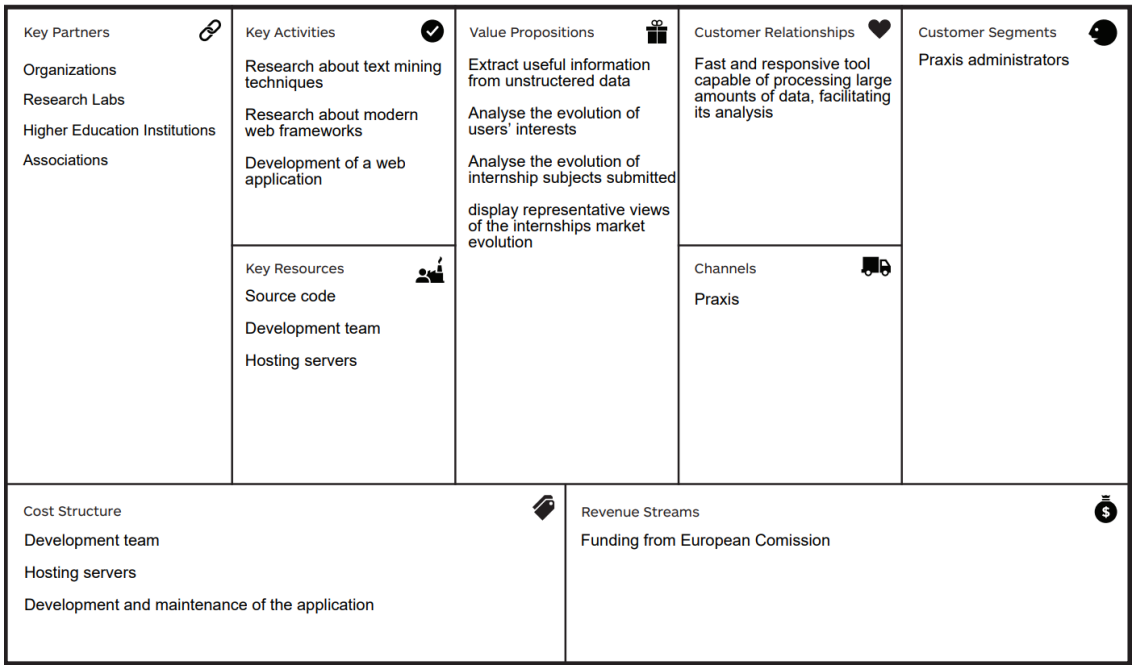

Figure 2.1: Business Model Canvas.

Chapter 3

State of the art

This chapter describes the technical knowledge needed to develop the solution, starting with a section dedicated to text mining, which focuses on three main topics: preprocessing techniques, word frequency and clustering.

The next section exposes the importance of data visualisation as a complement to text mining techniques.

Then, a section is dedicated to technologies in the domain of text mining, such as frameworks and libraries, that can be used in the development of this project.

Lastly, this chapter ends with the presentation of existing solutions related to the objectives of this thesis.

3.1 Text mining

Text mining is a specific form of data mining that deals with text. Also referred to as text analysis, text mining is the process of transforming unstructured text into meaningful and actionable information (MonkeyLearn 2020). By identifying topics, patterns, and relevant keywords, text mining obtains valuable insights without needing to go through all the data manually. Text mining allows organisations to analyse complex and large sets of data in a simple, fast and effective way, which leads to better data-driven business decisions.

Natural Language Processing (NLP) is a component of text mining that performs a special kind of linguistic analysis that essentially helps a machine "read" text (Expert System 2020). The ultimate goal of NLP is to derive meaning from human languages in a manner that is valuable. In linguistics, a corpus is usually used to refer to a large set of text documents.

Text mining has many applications. From governmental organisations, research institutions, and business needs, the uses of text mining are virtually endless. It is, for example, very useful in scientific research. Scientists communicate through scientific publications and it is estimated that over 50 million scientific publications exist (Jinha 2010), increasing at a rate of 2.5 million new articles being published every year (Ware and Mabe 2015). It is increasingly difficult for researchers to keep up with publications in their own field. Text

mining can solve this problem by finding information that would otherwise be difficult to find.

A text mining process (Figure 3.1) starts by gathering data from multiple sources, such as websites, databases, etc. The assembled data then goes through the preprocessing step, where the data is prepared and transformed. Once prepared, the next step is creating an index of certain terms. This allows the processed data to be quickly accessed. The data has been properly preprocessed and is now ready to be mined. At this point, one or a combination of techniques are used to extract meaningful information with the purpose of revealing new knowledge. The results from the mining step are then evaluated and visualised, so they can be interpreted by the user.

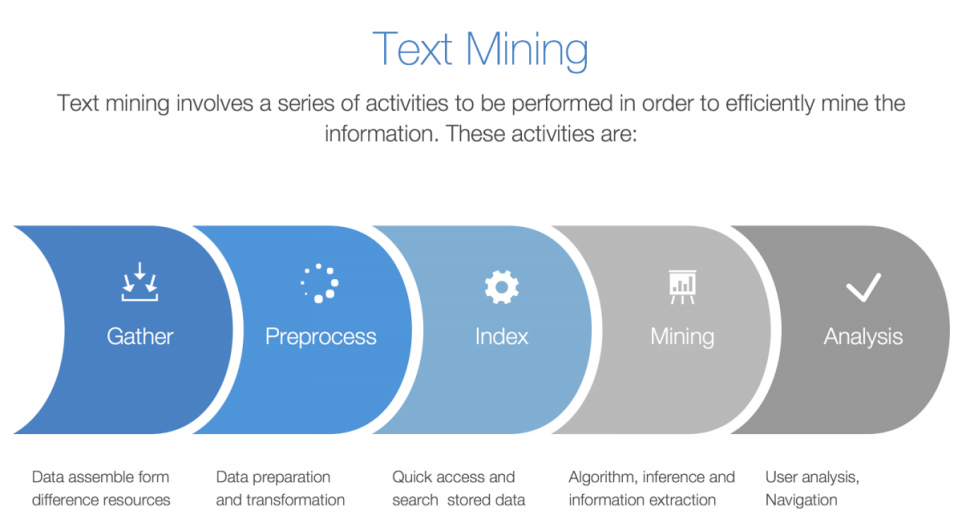

Figure 3.1: Text mining process steps. (Chang et al. 2018)

The upcoming sections present a detailed description of various text mining preprocessing and processing techniques.

3.1.1 Text Preprocessing

Text preprocessing is a key component in a text mining process, since it is responsible for preparing the data for the text mining processing techniques. The following sections present various preprocessing techniques.

Lowercasing

Lowercasing text data is one of the simplest but at the same time one of the most effective forms of text preprocessing. Although applicable to most text mining problems, preserving the uppercase might be relevant in certain situations, such as the identification of entities where capitalisation is an indication of high relevance.

Tokenization

Tokenization (Mayo 2017) is the process of splitting longer strings of text into smaller pieces called tokens. Larger chunks of text can be tokenized into sentences and sentences can be tokenized into words. Segmentation is usually used to refer to tokenizing larger chunks of text into paragraphs or sentences, while tokenization usually refers to the process that results exclusively in words.

Tokenization is language-specific, and each language has its own tokenization requirements. Most alphabetic languages follow relatively straightforward conventions to break up sentences and words, and so they are relatively simple to tokenize (Mohler 2019).

Stopword removal

Stopwords are a set of commonly used words in a language that frequently appear in text but do not have much content information (e.g. prepositions, conjunctions, etc.) (Allahyari et al. 2017). Removing the words that provide the lowest information puts the focus on the most important words instead.

Stopword lists can come from pre-established sets or be created according to a specific domain.

Stemming

Stemming (Jivani 2011) is the method used to identify the root/stem of a word and is usually done by removing any attached suffixes and prefixes (affixes). Morphological variants of words usually have similar semantic interpretations, so stemming looks to identify each word form with its base form. Thus, the key terms of a query or document are represented by stems rather than by the original words. This reduces the total number of distinct terms in a document or a query which in turn will reduce the processing time of the final output.

The stem is obtained after applying a set of rules but without bothering about the Part-Of-Speech (POS) or the context of the word occurrence, which means the stem may not be a real root word, but just a canonical form of the original word (Ganesan 2019). For example, the words "trouble", "troubled" and "troubles" might be converted to "troubl" instead of "trouble".

Lemmatization

Lemmatization (Jivani 2011) is very similar to stemming, since it also looks to identify the root of a word, in this case called lemma. The main difference is that lemmatization takes into consideration the POS and the context of the word in the given sentence. It actually transforms words to the actual root.

In this case, the words "trouble", "troubled" and "troubles" would actually be converted to "trouble".

3.1.2 Word Frequency

A central question in text mining is how to quantify what a document is about, therefore it is important to measure how important a word is within and across documents. Term frequency *tf* identifies how frequently a word occurs in a document (Silge and Robinson 2020). A bigger number of occurrences correspond to a bigger importance to the meaning of the document.

However, many common words that have no importance such as "the" and "for" usually top the term frequency list. As already mentioned previously, one approach to remove these low context words is to use a list of stopwords.

Another approach is to look at a term's inverse document frequency *idf*, which decreases the weight for commonly used words and increases the weight for words that are not used very much in a collection of documents. The *idf* is defined in formula 3.1 (Boehmke 2017).

$$idf(t, D) = log(\frac{N}{nt}) \quad (3.1)$$

Where the *idf* of a given term t in a set of documents D is a function of the total number of documents being assessed N and the number of documents in D where the term t appears nt. It is important to stress that while *tf* is a measure that depends exclusively on the document itself, *idf* depends also on the corpus where the document is included.

The *tf* and the *idf* statistics can be combined to calculate the frequency of a term adjusted for how rarely it is used, the *tf-idf* statistic. The *tf-idf* is defined in formula 3.2 (Boehmke 2017).

$$tf - idf(t, d, D) = tf(t, d) * idf(t, D) \quad (3.2)$$

Where *tf-idf* for a particular term t in document d for a set of documents D is the product of that term's *tf* and *idf* statistics.

3.1.3 Clustering

Clustering is one of the most popular data mining algorithms and has been extensively studied in the context of text (Allahyari et al. 2017). Clustering is the task of segmenting a corpus into partitions, where each group (cluster) consists of a number of similar documents. Its aim is to find intrinsic structures in information, and arrange them into significant subgroups for further study and analysis. Clustering is an unsupervised learning method since it tries to find hidden structure out of unlabeled data.

There are many clustering algorithms that can be used in the context of text mining. The following sections describe the clustering algorithms relevant to this thesis.

K-Means algorithm

Algorithm 3.1 presents the steps of a K-Means algorithm (Padmanabhuni and Bindu 2013). The K-Means algorithm is initiated by selecting a number of k objects randomly, where k represents number of desired clusters. These k objects are initialized as the initial cluster center points. The remaining objects are assigned to the clusters to which they are most similar, depending on the distance between the objects and the means of the clusters. The centroid of each cluster is updated based on the points assigned to the cluster. These steps are repeated for a set number of iterations or until the group centers don't change much between iterations.

Algorithm 3.1 K-Means algorithm

1: **Input**: D: a dataset of n objects, k: number of clusters
2: **Output**: A set of k clusters
3:
4: **procedure** K-Means
5: 1. Arbitrarily choose k objects from D as the initial cluster centers;
6: **repeat**
7: 2. Assign each object to the cluster to which the object is most similar, based on the mean value of the objects in the cluster;
8: 3. Update the cluster means, i.e., calculate the mean value of the objects for each cluster;
9: **until** no change;
10: **end procedure**

K-means is one of the most well-known clustering algorithms. Its popularity can be justified by its simplicity and efficiency, since all it does is computing the distances between points and group centers. It thus has a linear complexity $O(n)$, and the number of necessary iterations is usually quite small (Seif 2018).

Lingo algorithm

The Lingo algorithm first attempts to discover descriptive labels for future clusters and only then proceeds to assigning each cluster with matching documents. To find the labels, Lingo builds a term-document matrix for all input documents and decomposes the matrix to obtain a number of base vectors. Each vector gives rise to one cluster label. To complete the clustering process, each label is assigned documents that contain the label's words (Padmanabhuni and Bindu 2013).

The Lingo algorithm consists of 5 phases (Osinski and Weiss 2004), represented in Algorithm 3.2. The first phase is responsible for the preprocessing of input documents. In phase two, frequent terms and phrases are discovered in a combined set of all documents. Phase three uses Singular Value Decomposition (SVD) techniques to extract orthogonal vectors of the term-document matrix, that represent distinct topics in the input data. Phase four uses

the Vector Space Model (VSM) to assign the highest scoring documents as each cluster's content. The last phase is applying a score function to all clusters to sort them for display.

Algorithm 3.2 Lingo algorithm

```
1:  Input: D: a set of documents
2:  Output: A set of k clusters
3:
4:  procedure Lingo(Phase 1: Preprocessing)
5:      for all document d in D do
6:          perform text segmentation of d;
7:          identify language of d;
8:          apply stemming and mark stopwords in d;
9:      end for
10: end procedure
11: procedure Lingo(Phase 2: Frequent Phrase Extraction)
12:     discover frequent terms and phrases;
13: end procedure
14: procedure Lingo(Phase 3: Cluster Label Induction)
15:     Use a factorization matrix to create a CLuster Label Candidates set;
16: end procedure
17: procedure Lingo(Phase 4: Cluster Content Discovery)
18:     for all label l in CLuster Label Candidates do
19:         use the VSM (Vector Space Model) to determine the cluster contents;
20:     end for
21: end procedure
22: procedure Lingo(Phase 5: : Final Cluster Formation)
23:     calculate cluster scores;
24:     apply cluster merging;
25: end procedure
```

Suffix Tree Clustering (STC) algorithm

The STC algorithm groups the input documents according to the identical phrases they share. Phrases are used both to discover and to describe the resulting groups. Algorithm 3.3 presents the phases of the STC algorithm (Osinski 2003).

Algorithm 3.3 STC algorithm

```
1:  Input: D: a set of documents
2:  Output: A set of k clusters
3:
4:  procedure STC(Phase 1a: Creation of a Generalized Suffix Tree of all sentences)
5:      for all document d in D do
6:          for all sentence s in d do
7:              insert sentence and all its substrings into generalised suffix tree and update internal nodes with the
    index to current document while rearranging the tree;
8:          end for
9:      end for
10: end procedure
11: procedure STC(Phase 1b: Build a list of base clusters)
12:     for all node in tree do
13:         add a base cluster to the list of base clusters;
14:     end for
15: end procedure
16: procedure STC(Phase 2: Merge base clusters)
17:     build a graph where nodes are base clusters and there is a link between node A and B if and only if the number
    of common documents indexed by A and B is greater than the Merge_Threshold;
18:     clusters are coherent subgraphs of that graph;
19: end procedure
```

The first phase builds a generalised suffix tree of all documents' sentences using words as basic elements. After all sentences are processed, the tree nodes contain information about the documents in which particular phrases appear. Documents that share the same phrase are then grouped into base clusters. The final phase merges base clusters into final clusters.

3.2 Data visualisation

While text mining techniques are the core and undoubtedly the most important part of any text analysis system, data visualisation is also a key part in text analysis. It is much easier to understand data in a visual format as opposed to lines and lines of text and numbers.

Data visualisation is the graphical representation of data in a graph, chart, or other visual format (import.io 2019). This makes the data easier for the human brain to understand and therefore makes it easier to see and understand trends, outliers, and patterns within large datasets. Data visualisation tools and technologies are essential to analyse massive amounts of information and make data-driven decisions.

Data visualisation is not only important for data scientists and data analysts, it is used across all industries to increase sales with existing customers and target new markets and demographics for potential customers.

Depending on the data being modeled, and what its intended purpose is, there are many different visualisation types to use. Sometimes a simple graph is the most effective, whereas other times a more complex visualisation is needed to get the job done. Each type of data visualisation can be used in different way. Seeing how data trends over time and how frequently events happen over time are two of the most common uses of data visualisation, since most data has an element of time involved (Analytiks 2020).

3.2.1 Data-Driven Documents

Data-Driven Documents (D3) is a novel representation-transparent approach to visualisation for the web (Bostock, Ogievetsky, and Heer 2011). It is a JavaScript library for manipulating documents based on data, that produces dynamic, interactive data visualisations in web browsers. It makes use of Scalable Vector Graphics (SVG), HTML5, and Cascading Style Sheets (CSS) standards. D3's emphasis on web standards gives full capabilities of modern browsers, combining powerful visualisation components and a data-driven approach to DOM (Document Object Model) manipulation (D3 2020).

D3 provides many types of visualisations, such as the popular bar, line and pie charts. This facilitates the process of displaying the text mining results, in a way the user can analyse large sets of data much quicker through charts rather than looking at a spreadsheet.

3.2.2 Word clouds

A word cloud, also referred to as tag cloud, is a stunning visualisation format that highlights important textual data points, immediately conveying crucial information (Boostlabs 2014).

Word clouds are generally used to display how frequently words appear in a set of documents. The bigger and bolder the word appears, the more often it's mentioned within a given text and the more important it is.

Figure 3.2 presents an example of a word cloud, showcasing how simple and informative word clouds can be to convey insights to the end user.

Figure 3.2: Word cloud example

In this thesis, word clouds are going to be specifically useful to display the most searched keywords by the users and the most present keywords in offers submitted in the Praxis website.

3.3 Technologies

This section presents technologies, such as frameworks and libraries, in the domain of text mining that might be useful for the development of this project, as well as a method that helped the selection of the technologies to use.

3.3.1 NTLK - Natural Language Toolkit

NLTK (Bird, Klein, and Loper 2009) is a free, open-source platform for building Python programs to work with human language data. The library was developed by Steven Bird and Edward Loper at the University of Pennsylvania and played a key role in breakthrough NLP

research. It is one of the most powerful NLP libraries which contains numerous packages that provide numerous NLP tasks.

NLTK provides many preprocessing techniques such as tokenization, stemming, lemmatization, tagging and parsing. It also includes a predefined list of stopwords in many different languages.

NLTK also provides tasks for frequency distributions, text classification and clustering. Clustering includes the algorithms: K-means, E-M and Group Group Average Agglomerative Clusterer (GAAC).

3.3.2 spaCy

spaCy (spaCy 2020) is a free, open-source library for advanced Natural Language Processing in Python. spaCy is designed specifically for production use and helps build applications that process and understand large volumes of text. It can be used to build information extraction or natural language understanding systems, or to preprocess text for deep learning.

It includes features such as tokenization, part-of-speech tagging, lemmatization, stopwords and text classification.

3.3.3 Scikit-learn

Scikit-learn (scikit-learn 2020) is an open source machine learning library for Python that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.

Scikit-learn provides a set of tools to work with text data, which include tokenization, stopwords and frequency distributions. It also provides a module dedicated to clustering unlabeled data, which has many clustering algorithms ready to use, like K-Means, hierarchical clustering and agglomerative clustering.

3.3.4 Carrot2

Carrot2 (Carrot Search 2020) is a programming library for clustering text implemented in Java. It can automatically discover groups of related documents and label them with short key terms or phrases.

Carrot2 comes with a suite of tools and APIs such as the Carrot2 Document Clustering Server (DCS), which exposes Carrot2 clustering as a REST service. This means Carrot2 can easily be integrated with languages other than Java.

Currently, Carrot2 offers 3 clustering algorithms: Lingo, STC and K-Means. The algorithms differ in terms of the main clustering principle and hence have different quality and performance characteristics. Figure 3.3 summarises key characteristics of each algorithm shipped in Carrot2.

A comparison of key characteristics of Carrot^2 clustering algorithms.

Feature	Lingo	STC	k-means
Cluster diversity	**High**, many small (outlier) clusters highlighted	**Low**, small (outlier) clusters rarely highlighted	**Low**, small (outlier) clusters rarely highlighted
Cluster labels	**Longer**, often more descriptive	**Shorter**, but still appropriate	**One-word only**, may not always describe all documents in the cluster
Scalability	**Low.** For more than about 1000 documents, Lingo clustering will take a long time and large memory .	**High**	**Low**, based on similar data structures as Lingo.
Overlapping clusters	**Yes.** A document can belong to more than one cluster.	**Yes.** A document can belong to more than one cluster.	**No.** A document can belong to only one cluster.

Figure 3.3: Key characteristics of Carrot2 clustering algorithms (Carrot Search 2020).

While the table above can be useful to determine which algorithm to choose, the ultimate judgment should be based on an empirical evaluation with a specific set of documents.

3.3.5 R

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It provides an effective data handling and storage facility, a large, coherent, integrated collection of intermediate tools for data analysis and graphical facilities for data analysis (The R Foundation 2020).

R can be extended via packages available through the CRAN family of Internet sites covering a very wide range of modern statistics.

The tm package is an R package that provides a text mining framework. It includes methods for data import, corpus handling, preprocessing, metadata management, and creation of term-document matrices (Feinerer 2019).

Corpus is the main structure for managing documents, it represents a collection of text documents. Metadata is used to annotate text documents with additional information. Term-document matrices are matrices created from a corpus (terms as rows and documents as columns, or vice versa). Many R functions like clustering or classification can be applied to these matrices.

3.3.6 Technologies selection

After researching frameworks and libraries in the domain of text mining, the next step is selecting which ones suit the best for the development of the project.

The Analytic Hierarchy Process (AHP) is a method developed by Thomas L. Saaty in the 1970s for organising and analysing complex decisions, using math and psychology. AHP provides a rational framework for a needed decision by quantifying its criteria and alternative options, and for relating those elements to the overall goal (Passage Technology 2020). So, it is advantageous to use this mechanism to compare and select the technologies to use in this project.

The following criteria were defined as the main relevant points to choose the technologies to use: language the technology uses, available documentation and the amount/quality of the preprocessing and clustering tasks the technology provides. The values attributed to each criteria are the result of the research previously conducted on the various technologies.

Table 3.1 shows the weight attributed to each criteria. Preprocessing and clustering tasks is the criteria with the most importance (0,7014), followed by the documentation (0,2132) and finally the language (0,0853).

Table 3.1: AHP - criteria weight attribution.

	Language	**Documentation**	**Preprocessing and Clustering tasks**	**Weight**
Language	1	$\frac{1}{3}$	$\frac{1}{7}$	0,0853
Documentation	3	1	$\frac{1}{4}$	0,2132
Preprocessing and Clustering tasks	7	4	1	0,7014
Total	11,0000	5,3333	1,3929	1

The research conducted above on technologies suggests that Python and R are the two most suitable programming languages for the development of this project. Ultimately, it was decided to use Python over R, since it provides better features not only in terms of text mining, but also in terms of database connection, API setup, etc. So the AHP method is not including R.

Table 3.2: AHP - technology selection based on preprocessing tasks.

	Language	Documentation	Preprocessing tasks	Priority
NLTK	8	9	10	0,3974
spaCy	8	9	8	0,3390
Scikit-learn	8	7	6	0,2635
Total	24	27	24	1

Table 3.3: AHP - technology selection based on clustering tasks.

	Language	Documentation	Clustering tasks	Priority
Carrot2	10	10	10	0,3914
Scikit-learn	8	9	10	0,3760
NLTK	8	5	6	0,2325
Total	26	24	26	1

Table 3.2 suggests that both NLTK and spaCy are a good choice when it comes to preprocessing tasks, however NLTK has a slight advantage when it comes to the preprocessing tasks it provides. Table 3.3 suggests that both Carrot2 and Scikit-learn are a good choice when it comes to clustering tasks, however Carrot2 comes on top due to a better documentation and the freedom of using any language due to the REST service it provides.

So, for this thesis, NLTK is going to be used to handle preprocessing tasks and Carrot2 to handle clustering tasks, since according to the weights attributed to each criteria, both present the highest score in the AHP method.

3.4 Existing solutions

This section presents existing solutions in the market that present features related to the objectives of this thesis.

3.4.1 Google Trends

Google Trends (Figure 3.4) (Google 2020) is a search trends feature that shows how frequently a given search term is entered into Google's search engine relative to the site's total search volume over a given period of time. It gives insights on the popularity of keywords and when they were trending during a certain time period, which helps understand how trends change over time.

Another useful feature is the comparison of keywords. Users can enter one or more keywords and view their relative popularity. It is also possible to filter the results by location, time frame, category and type of search (web, image, news, shopping, Youtube).

Additionally, it shows how search interests vary by regions and suggests related keywords that compliment an original search query.

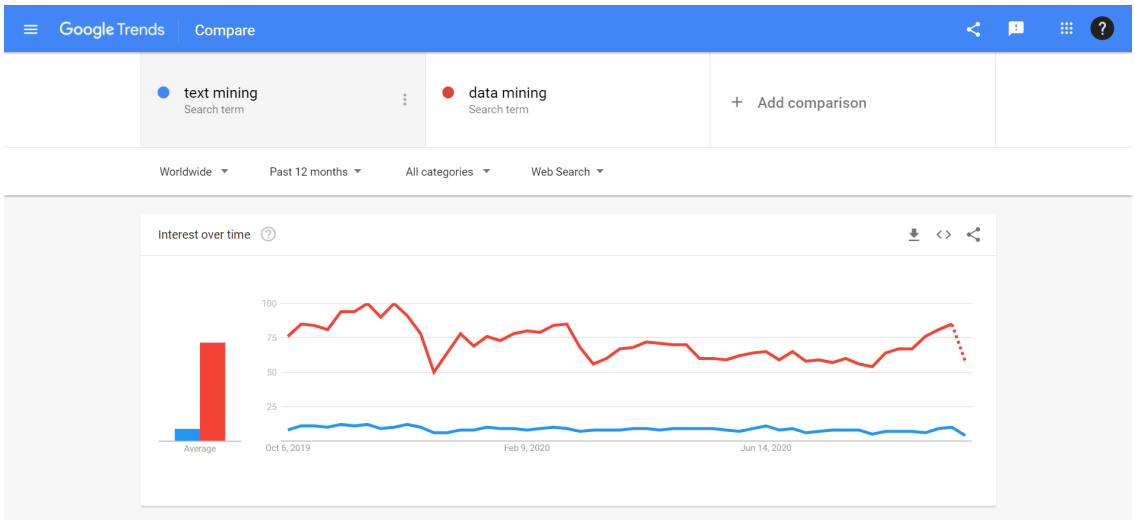

Figure 3.4: Google Trends.

3.4.2 WordStream Keyword Tool

WordStream Keyword Tool (Figure 3.5) (WordStream 2020) is a keyword research tool that utilises the latest Google and Microsoft's Bing search engines search data.

There is the possibility to either enter a keyword or a URL of a webpage to see keyword data for that page. To maximise the relevance of the results, it is also possible to filter the results by industry and country.

After entering the initial search query, a list of related keyword suggestions is presented, as well as their search volume on Google and Bing.

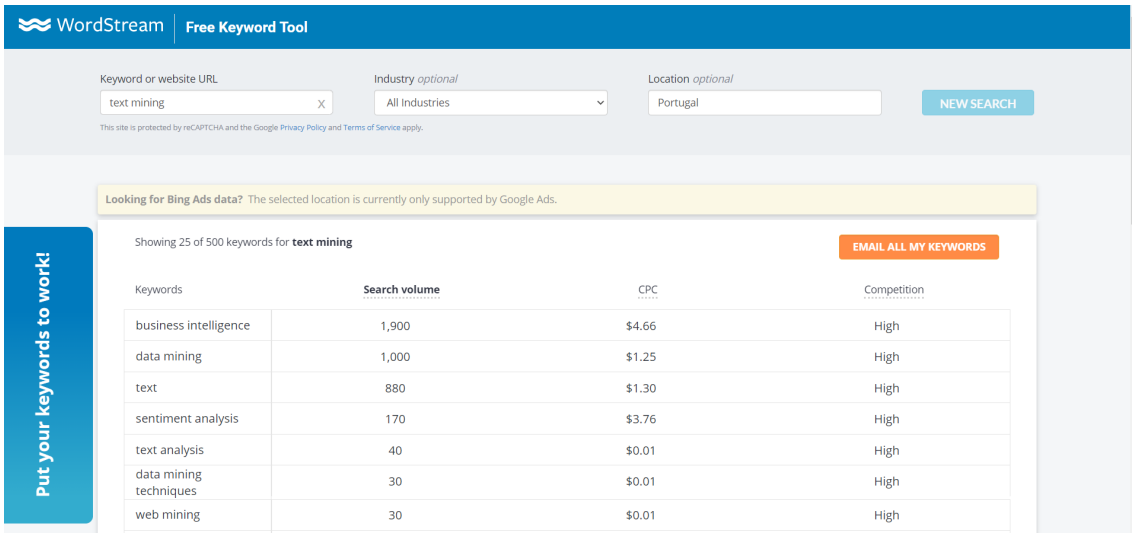

Figure 3.5: WordStream Keyword Tool.

3.4.3 Carrot2 search results clustering engine

Carrot2 (Carrot2 2020) search results clustering engine (Figure 3.6) is a live demo implementation of the Carrot2 DCS. It applies clustering to web search results provided by the eTools search engine and to medical documents from the PubMed database of medical abstracts.

After entering a keyword, the search results are organised into clusters, that are displayed through either folders, a treemap or pie chart. Clicking on a cluster shows the search results contained in that cluster.

The results can be filtered by language and country, and it is also possible to choose which clustering algorithm to use, between Lingo, STC and K-Means.

As stated in the previous section, Carrot2 is going to be used in this thesis to handle document clustering.

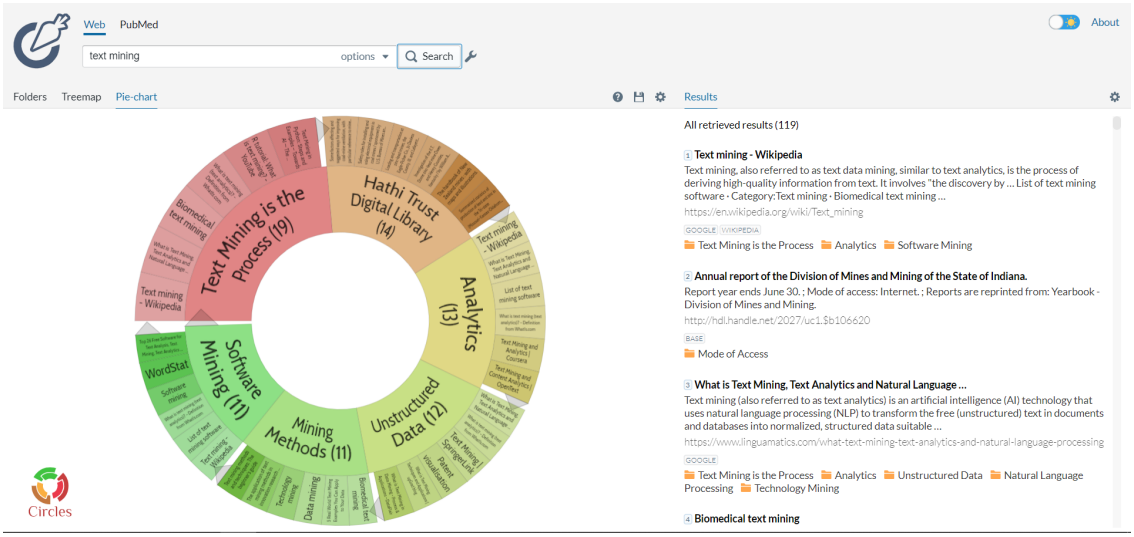

Figure 3.6: Carrot2 search results clustering engine.

Chapter 4

Analysis and Design

This chapter starts by defining the functional requirements the system should meet, followed by an analysis of all the data available in the already existing Praxis database.

The next section presents two architecture proposals for the system, followed by the description of the selection process used to select the final architecture.

Finally, the last section presents a high-level view of the various processes in the system.

4.1 Functional requirements

Functional requirements define the basic system behaviour, how the system responds to inputs. Figure 4.1 represents a Unified Modeling Language (UML) use case diagram, that specifies how the actor interacts with the system without worrying about implementation details.

The main and only actor that interacts with the system is the Praxis Administrator, as this is the person responsible for analysing and interpreting the data displayed by the tool. The system consists of the following functional requirements:

- View general stats
- Calculate and view the frequency of keywords in searches and offers
- View the evolution of a keyword
- Generate and view clusters for a set of offers

Table 4.1 presents a detailed description of each use case.

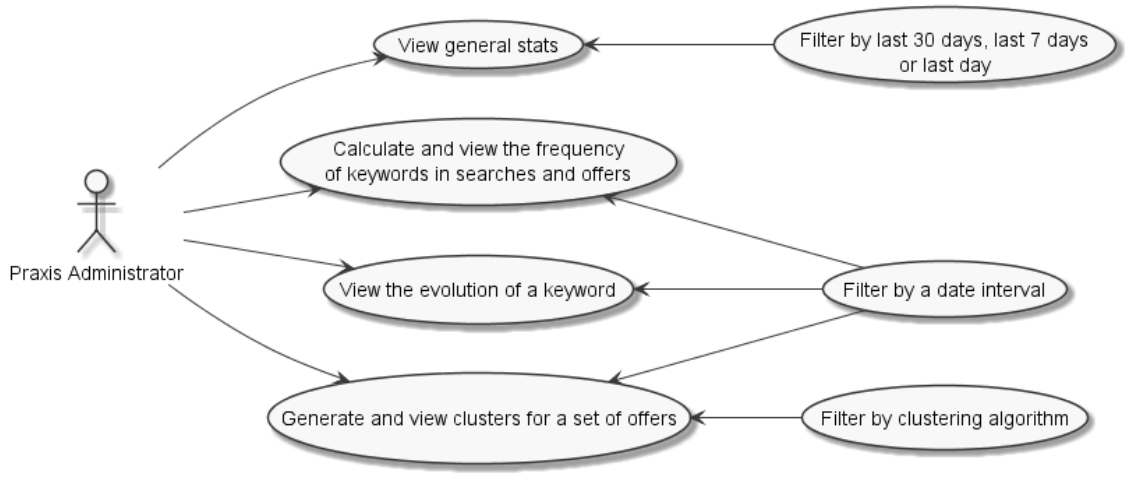

Figure 4.1: UML use case diagram for the Praxis Administrator.

Table 4.1: Detailed description of each use case.

Use Case	**Description**
View general stats	The Praxis Administrator can view general statistics about the Praxis platform. This includes, for either the last 30 days, last 7 days or last day, the number of searches, the number of offers, the number of active offers, the average number of results per search, the number of student logins and the number of provider logins.
Calculate and view the frequency of keywords in searches and offers	Given a date interval, the Praxis Administrator can view the most searched keywords and the most common keywords in offers' titles through word clouds.
View the evolution of a keyword	For a date interval, the Praxis Administrator can view the number of occurrences of a keyword for each date in between the date interval.
Generate and view clusters for a set of offers	Upon selecting the clustering algorithm and the date interval, the Praxis Administrator can view the clusters generated for the set of offers in the date interval.

4.2 Praxis database analysis

The current Praxis database is implemented in MySQL and is composed by 58 tables, that store data from 2014 up to 2020. Only 3 tables are relevant for the development of this project: Proposal, SearchRequest and fos_user, all represented in Figure 4.2. For simplicity purposes, only the relevant columns are displayed for each table.

The Proposal table contains all the offers submitted through the Praxis website. The columns title and description are easily the most important columns in this table, since

they provide all the information regarding what kind of offers are being submitted. The column submitDate represents the submission date of an offer, which is useful to filter offers within a date interval. The columns status and validTo are also useful to filter the analysis to only active offers if there is a need to do so. As of right now, the proposal table has a total of 2,251 offers, to which only 60 are active.

The SearchRequest table contains all the searches made on the Praxis website by its users. The column query is undoubtedly the most important in this table, since it represents a set of one or more keywords the users type to search for results. These keywords provide information about what kind of offers the users are currently looking for. By combining this information with the date of when the search was made, it is possible to track the users' interests over time. The column nResults shows the number of results presented to the user in response to a search, from which it is possible to depict which interests are being satisfied and which are not. As of right now, the table SearchRequest holds a total of 636,560 searches, from which 25% are keyword searches.

The fos_user table contains all the information regarding every user registered on the Praxis platform. Each registered user has an associated role, that can be admin, provider or student. The column last_login is particularly useful to see how many users are using the platform, for example, on a daily basis. There are a total of 10,240 registered users on the Praxis platform at the moment, where 7% are providers and 93% are students.

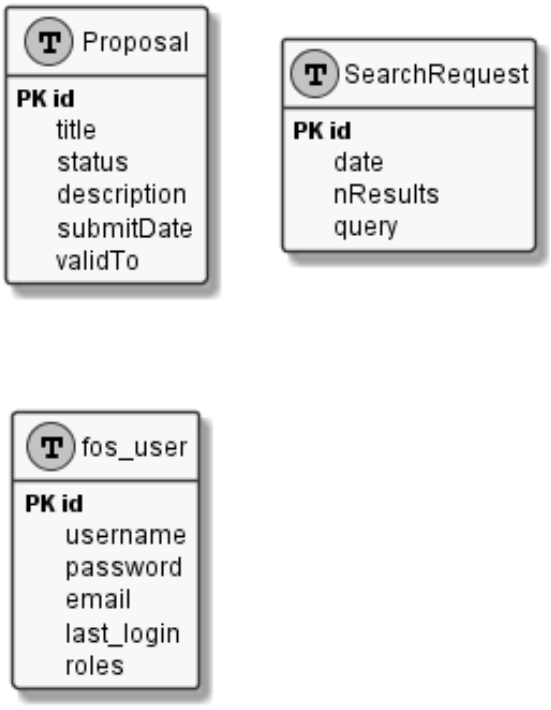

Figure 4.2: Praxis database tables.

4.3 Project's architecture

This section presents two architecture proposals to develop the Praxis Market Drift solution.

4.3.1 Architecture 1

Figure 4.3 presents the proposed architecture for the system do develop, a simplified component diagram showing how the different components interact with each other.

The Praxis MySQL database is the only component that already exists, and contains all the data regarding searches and offers on the Praxis platform.

The Backend is responsible for fetching searches and offers data from the Praxis MySQL database. The data then goes through an appropriate preprocessing so that text mining techniques can be applied.

The Carrot2 DCS is responsible for generating clusters in response to a set of offers sent by the Backend.

The Frontend is the user interface. It displays the data coming from the Backend according to the user's specified task.

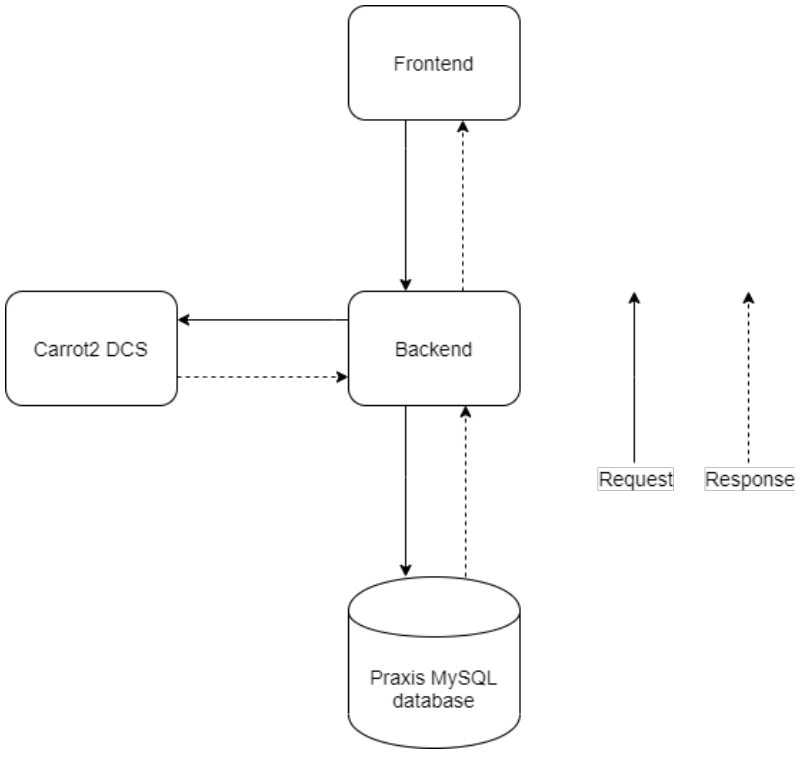

Figure 4.3: Proposed architecture 1.

4.3.2 Architecture 2

Figure 4.4 presents the proposed architecture for the system do develop, a simplified component diagram showing how the different components interact with each other.

The Praxis MySQL database is the only component that already exists, and contains all the data regarding searches and offers on the Praxis platform.

The Preprocessing engine is responsible for fetching searches and offers data from the Praxis MySQL database and apply an appropriate preprocessing to the data. The preprocessed data is then stored into the Preprocessed data MySQL database.

The Backend is responsible for fetching data from the Preprocessed data MySQL database. It then applies text mining techniques to that data.

The Carrot2 DCS is responsible for generating clusters in response to a set of offers sent by the Backend.

The Frontend is the user interface. It displays the data coming from the Backend according to the user's specified task.

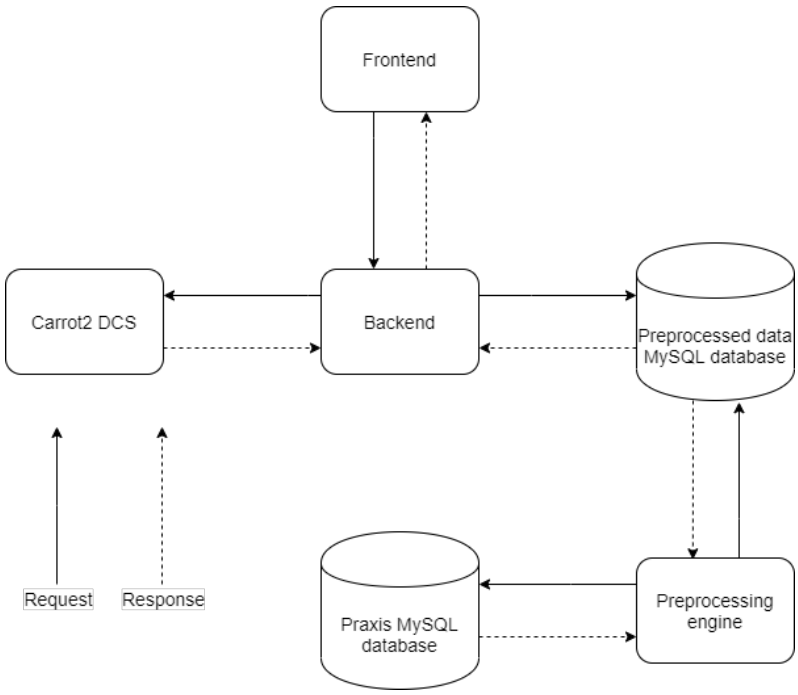

Figure 4.4: Proposed architecture 2.

4.3.3 Architecture selection

The difference between the 2 proposed architectures is in the database the backend fetches data from.

In the first, the backend fetches data directly from the Praxis MySQL database. In the second, there are 2 additional components, the Preprocessing engine, responsible for the preprocessing of the data, and the Preprocessed data MySQL database, to store the preprocessed data. The Backend then fetches data from the Preprocessed data MySQL database.

The idea behind the second architecture was to have a database that would store only the relevant preprocessed data to this project, meaning there would be no need to access the official Praxis database and apply preprocessing to the data every time a request is made.

However, since the system is not going to deal with huge amounts of data, and since the data in the Praxis database is already decently organised, it was ultimately decided to go with architecture 1, which also requires less development time because it has less components.

The system is still going to be built keeping in mind that it should easily be upgradable to architecture 2 if there is a need to do so in the future.

4.4 Processes' flow

The following sections present sequence diagrams that showcase high-level interactions between the various components of the system.

4.4.1 View general stats

Figure 4.5 presents the sequence of requests needed to view the general statistics.

The Frontend starts by sending a GET request to the Backend, which then queries the database for the data. The Backend then returns the data sent by the database in an appropriate format and the Frontend presents the data to the Praxis Administrator.

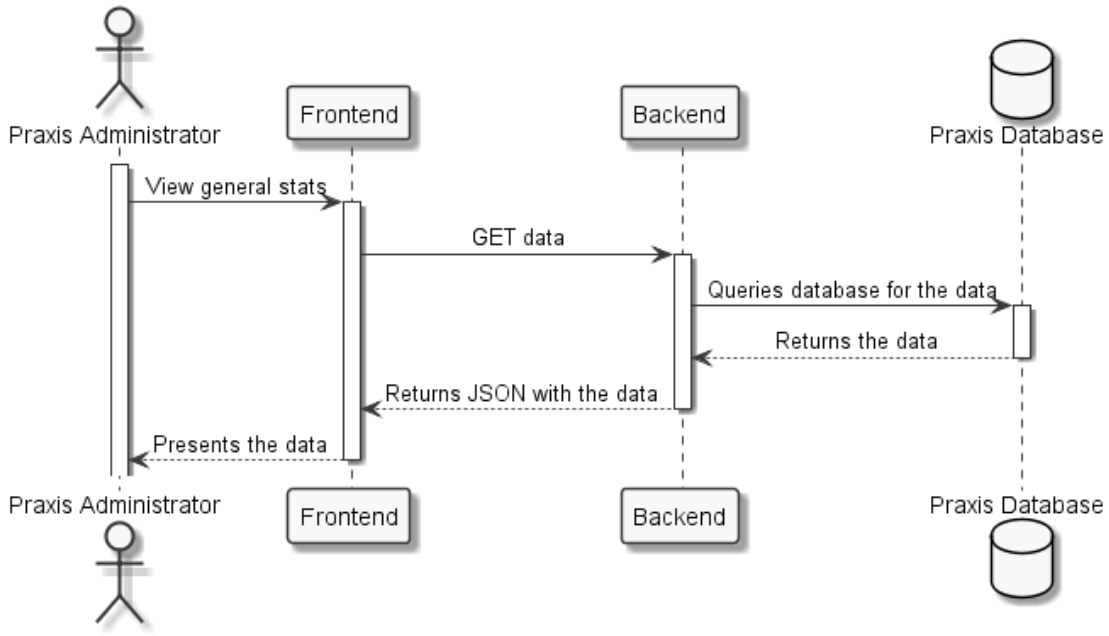

Figure 4.5: View general stats sequence diagram.

4.4.2 View searches keyword frequency

Figure 4.6 presents the sequence of requests needed to view the frequency of keywords in searches.

Upon selecting a date interval, a GET request for the frequency of keywords is sent to the Backend. The Backend then queries the database for all the searches in the selected date interval, applies an appropriate preprocessing to the data and calculates the frequency of each keyword. The Frontend then presents the results to the Praxis Administrator through a word cloud.

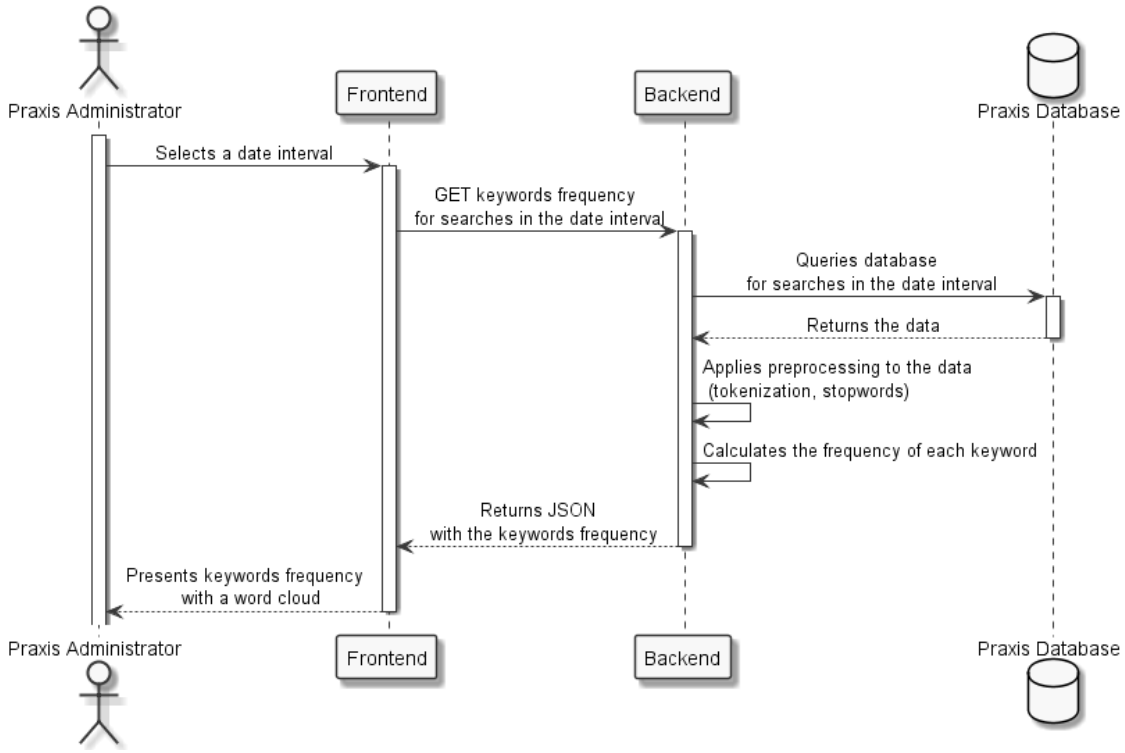

Figure 4.6: View searches keyword frequency sequence diagram.

4.4.3 View clusters

Figure 4.7 presents the sequence of requests needed to view the generated clusters for a set of offers.

Upon selecting a date interval and the clustering algorithm, a GET request to generate clusters is sent to the Backend. The Backend then queries the database for all the offers in the selected date interval and sends a POST request with all the offers and with the clustering algorithm to the Carrot2 DCS. The Carrot2 DCS returns the clusters generated for that set of offers, which are then presented to the Praxis Administrator by the Frontend through doughnut charts.

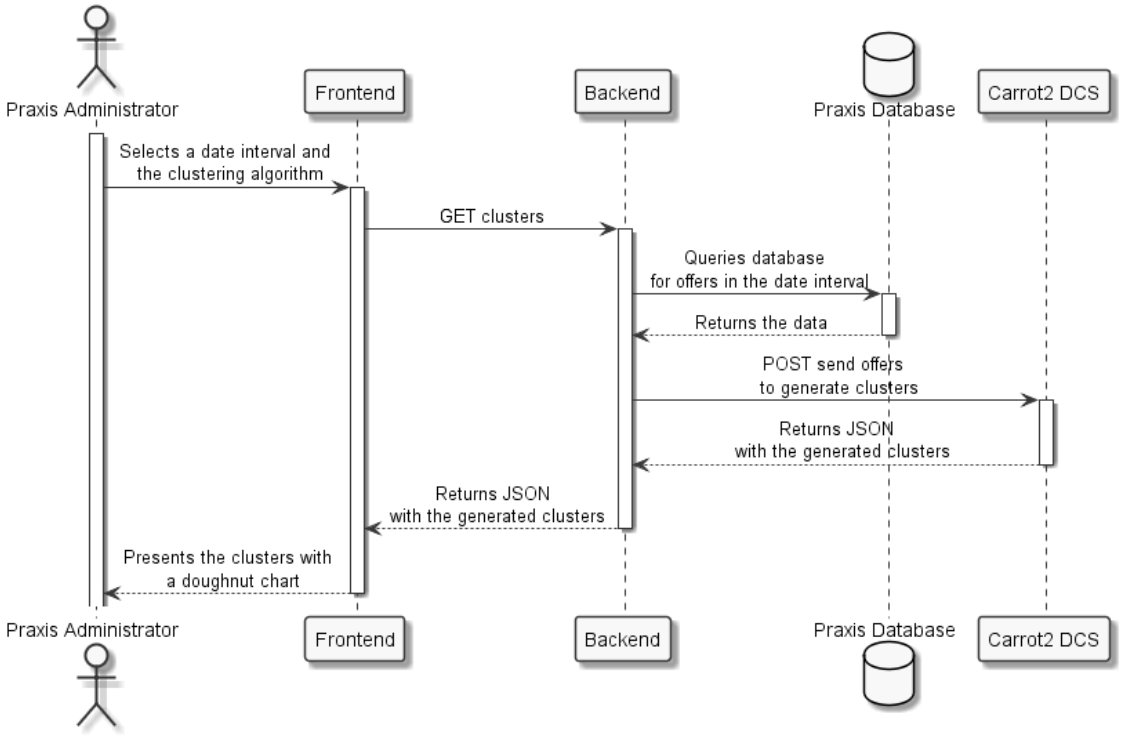

Figure 4.7: View clusters sequence diagram.

Chapter 5

Implementation

This chapter describes all the steps taken to implement each one of the components defined in the selected architecture in Section 4.3.

The first section details the process of setting up a local development environment to implement the solution using Docker containers.

The following sections present the implementation of each component, demonstrating how the application of text mining techniques enhance the quality of the information displayed to the user.

5.1 Setting up the development environment

Setting up a local development environment is the first step to start implementing the solution. Its frequent when developing, and sometimes testing, that the code works fine locally but when deployed into production, the local development environment is sufficiently different that the code fails. Differences in libraries, versions and operating systems are usually the root cause for this problem, since they are enough to break code that runs perfectly locally.

This particular problem inspired the usage of containers to setup the local development environment. Docker (Docker 2020a) is an open platform for developing, shipping, and running applications. Docker provides the ability to package and run an application in a loosely isolated environment called a container. A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. Figure 5.1 shows how Docker containerizes applications.

Each container needs an image to run. An image is a read-only template with instructions for creating a container and is often based on another image with additional customisation, the application as well as the configuration details needed to make the application run. A container is then a runnable instance of an image.

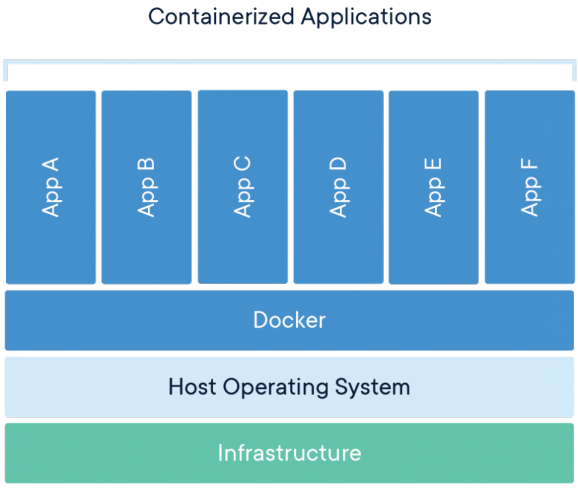

Figure 5.1: Docker containerized applications (Docker 2020a).

In order to setup the development environment, a container is needed for each one of the components defined in the selected architecture. Docker Compose is a tool for defining and running multi-container Docker applications (Docker 2020b). Compose uses a YAML Ain't Markup Language (YAML) file to configure the application's services, which then all can be created and started with a single command from the configuration.

Listing 5.1 presents the docker-compose.yml file used to setup the local development environment. The file defines 4 services and their respective configurations. The services can either use already existing images or images built from a Dockerfile. Listing 5.2 shows an example Dockerfile, in this case the one used for the backend service. The image contains all the dependencies the Python application requires, including Python itself.

```
version: '3.8'
services:
    frontend:
        stdin_open: true
        build:
            context: ./tmdei-frontend
            dockerfile: Dockerfile.dev
        ports:
            - '3000:3000'
        volumes:
            - '/app/node_modules'
            - './tmdei-frontend:/app'
        environment:
            - CHOKIDAR_USEPOLLING=true
        container_name: react_app
    backend:
```

```
        build:
            context: ./tmdei-backend
            dockerfile: Dockerfile.dev
        ports:
            - '5000:5000'
        volumes:
            - './tmdei-backend:/app'
        container_name: flask_app
    db:
        image: mysql
        environment:
            MYSQL_ROOT_PASSWORD: admin
        ports:
            - '3306:3306'
        volumes:
            - './db:/docker-entrypoint-initdb.d'
        container_name: mysql_server
    carrot2:
        build:
            context: ./carrot2
            dockerfile: Dockerfile.dev
        ports:
            - '8080:8080'
        container_name: carrot2_server
```

Listing 5.1: docker-compose.yml file

```
FROM python:alpine

RUN apk add build-base

WORKDIR /app

COPY . ./

RUN pip install -r requirements.txt

RUN python -m nltk.downloader punkt stopwords

CMD ["python", "app.py"]
```

Listing 5.2: Dockerfile for the backend service

The ports are mapped in the HOST:CONTAINER format. Using the frontend service as an example, the port 3000 in the container is being exposed and is reachable on the port 3000 in the host machine. Volumes are used to share data between containers and the host machine. The easiest way of defining a volume is through absolute path mapping, by defining the path on the host machine and mapping it to the container using the

HOST_PATH:CONTAINER_PATH format. If something is changed in the host machine, that change is reflected in the container without the need of restarting it.

Another advantage of Compose is that it sets up a single network for the app. Each container for a service joins the default network and is both reachable by other containers on that network, and discoverable by them at a hostname identical to the service name. For example, the backend container can connect to the URL http://db:3306 and start using the database server.

5.2 Developing the backend

According to the requirements defined in Chapter 4, the backend must connect and fetch data from the database, apply various text mining techniques, and connect to the Carrot2 DCS, to send a set of documents and receive a set of clusters.

5.2.1 Setting up a REST API

The first step to start developing the backend is setting up an Application Programming Interface (API), so the frontend can communicate with the backend and receive data. Flask (Flask 2020) is a lightweight Web Server Gateway Interface (WSGI) Python web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. Flask-RESTful (Flask-RESTful 2020) is an extension for Flask that adds support for quickly building Representational State Transfer (REST) APIs.

The main building block provided by Flask-RESTful are resources. Resources give access to multiple Hypertext Transfer Protocol (HTTP) methods just by defining methods on a resource. After creating a resource, the next step is adding it do the API, for which Flask-RESTful provides an add_resource() method. When it comes to handling endpoints, one or more URLs can be routed to a resource.

Mysql-connector-python (MySQL 2020) is a package that enables access to MySQL databases, using an API that is compliant with the Python Database API Specification. The connect() constructor creates a connection to the MySQL server and returns a MySQLConnection object. This package was used to connect to the Praxis MySQL database.

Listings 5.3 and 5.4 show an example on how to setup the REST API, create a resource, connect to the MySQL database and define the endpoints for a resource.

```
from flask import Flask
from flask_restful import Api
from flask_jwt_extended import JWTManager
from resources.searchesKeywordFrequency import SearchesKeywordFrequency

app = Flask(__name__)
api = Api(app)
```

```
app.config['JWT_SECRET_KEY'] = b'_5#y2L"F4Q8z\n\xec]/'
jwt = JWTManager(app)

api.add_resource(SearchesKeywordFrequency, "/api/searches/
    keywordFrequency/<string:fromDate>/<string:toDate>/<int:keywordNumber
    >",
                 "/api/searches/keywordFrequency/<int:lastDays>/<int:
    keywordNumber>")

if __name__ == '__main__':
    app.run()
```

Listing 5.3: API setup example

```
from flask_restful import Resource
from mysql import connector
from resources.dbconfig import config
from mining.keywordFrequency import keywordFrequency
from utils.utils import getDate, toJson

class SearchesKeywordFrequency(Resource):
    def get(self, lastDays = None, fromDate = None, toDate = None,
    keywordNumber = None):
        db = connector.connect(**config)
        cursor = db.cursor(dictionary=True)
        query = ''
        if lastDays:
            query = 'select query ' \
                    'from SearchRequest ' \
                    'where DATEDIFF("%s",date) between 0 and %i;' % (
    getDate(), lastDays)
        else:
            query = 'select query ' \
                    'from SearchRequest ' \
                    'where date between "%s" and "%s";' % (fromDate,
    toDate)
        cursor.execute(query)
        results = cursor.fetchall()
        cursor.close()
        db.close()
        fd = keywordFrequency(results, 'query')
        return toJson(fd.most_common(keywordNumber))
```

Listing 5.4: Resource example

5.2.2 Text mining with NTLK

To calculate the frequency distribution of keywords in searches and offers, the data fetched from the database needs to first be tokenized. Tokenization (Section 3.1.1) is the process of breaking complex strings into words and punctuation, called tokens.

The next step is defining a list of stopwords (Section 3.1.1). These are words do not provide any meaning and should be removed from the final results, like prepositions, conjunctions and punctuation. NLTK includes a corpus module that has a list of stopwords stored in 16 different languages.

The final step is to filter out the stopwords in the tokenized list and calculate the number of occurrences for each word. NLTK includes a probability module that provides a FreqDist() class that returns a list with the frequency of each word.

Listing 5.5 showcases the process to calculate the frequency distribution of keywords with NLTK modules.

```
import nltk
from mining.stopwordList import stopwordList
from utils.utils import toString

def keywordFrequency(keywords, column):
    words = toString(keywords, column)
    tokenized = nltk.word_tokenize(words)
    stopwords = stopwordList()
    return nltk.FreqDist(w for w in tokenized if w not in stopwords)
```

Listing 5.5: Keyword frequency with NLTK

5.2.3 Using the Carrot2 DCS

The Carrot2 DCS is essentially a single, stateless endpoint accepting JSON requests and returning JSON responses. A full clustering request is represented in Table 5.1.

The request body is composed by 3 elements: language, algorithm and an array of documents. Language and algorithm need to be specified, so that an appropriate preprocessing is applied to input text before clustering. There are 3 choices for the algorithm: Lingo, STC and K-Means. Documents for clustering are composed of one or more fields, where each field is a pair consisting of an identifier (name of the field) and value (a string or an array of strings). Once the request body is ready, the request must be sent using HTTP POST method to the /service/cluster endpoint.

The response is a list of clusters, where each cluster is composed by a description label, an array of documents contained in the cluster, an array of subclusters (if the algorithm supports hierarchical clustering) and a quality score.

Table 5.1: Clustering service request.

HTTP method	POST
Endpoint	/service/cluster

Request body example	{
	"language": "English",
	"algorithm": "Lingo",
	"documents": [
	"field": "foo bar" ,
	"field": "bar" ,
	"field": "baz"
	]
	}

Response example	{
	"clusters" : [
	{
	"labels" : [
	"Bar"
	],
	"documents" : [
	0,
	1
	],
	"clusters" : [ ],
	"score" : 0.09746237168594427
	}
	]
	}

5.2.4 Endpoints

Once everything is setup, the backend is then ready to accept requests. The following tables present the main endpoints provided by the developed backend. The full API specification can be consulted in Appendix A.

Table 5.2 represents a request to get a list of the most searched keywords by the users on the Praxis platform. This request can be sent to two different endpoints. Each endpoint requires a set of arguments used to filter the keywords accordingly. The first requires a date interval, while the second requires a value, which can take the values 30, 7 and 1, that correspond respectively to "Last 30 days", "Last 7 days" and "Last day". Both endpoints also require the number of keywords that should be returned in the response.

Table 5.2: Searches keyword frequency request.

HTTP method	GET
Endpoint	/api/searches/keywordFrequency/<string:fromDate>/<string:toDate>/<int:keywordNumber> /api/searches/keywordFrequency/<int:lastDays>/<int:keywordNumber>
Description	Returns searches keyword frequencies for the last specified days or for a date interval fromDate: start date (yyyy-mm-dd) toDate: end date (yyyy-mm-dd) keywordNumber: max number of keywords to return Possible values for lastDays: 30, 7, 1

Response example	[
	{
	"text": "engineering",
	"value": 349
	}
	]

Table 5.3 represents a request to get the evolution of a specific keyword. The endpoint requires a date interval and a keyword as arguments. The response returns a list with the number of occurrences for that keyword for each date between the specified date interval.

Table 5.3: Searches keyword evolution request.

HTTP method	GET
Endpoint	/api/searches/keywordEvolution/<string:fromDate>/<string:toDate>/<string:keyword>
Description	Returns the number of occurrences of a keyword in searches for each date in the date interval fromDate: start date (yyyy-mm-dd) toDate: end date (yyyy-mm-dd) keyword: the name of the keyword

Response example	[
	{
	"keywordCount": 2,
	"date": "2020-01-01",
	}
	]

Table 5.4 represents a request to get a list of clusters. The endpoint requires a date interval and the clustering algorithm as arguments. After obtaining all the offers in the specified date interval, a request is sent to the Carrot2 clustering service, which returns the list of clusters in response.

Table 5.4: Clusters request.

HTTP method	GET
Endpoint	/api/clusters/<string:fromDate>/<string:toDate>/<string:algorithm>
Description	Returns clusters for a set of offers in the date interval fromDate: start date (yyyy-mm-dd) toDate: end date (yyyy-mm-dd) Possible values for algorithm: lingo, stc, bkmeans

Response example	{
	"clusters" : [
	{
	"labels" : [
	"Graphic Design"
	],
	"documents" : [
	{
	"id": 2263,
	"title": "Graphic Designer / Researcher"
	},
	],
	"clusters" : [ ],
	"score" : 34.56174212000691
	}
	]
	}

5.3 Developing the frontend

The frontend plays a crucial role in every single application, since its role is to display the information in the best way possible. The quality of a frontend can make or break the first impression of the end user. So, the goal of this application is to provide an interactive tool that displays quality information that can be directly interacted with. The user can then analyse this data accordingly and get better insights on the evolution of the Praxis market.

With these requirements in mind, React was the framework of choice to develop the frontend. React (React 2020) is a JavaScript library created for building fast and interactive

user interfaces. React combines basic HTML and JavaScript concepts with some beneficial additions and is component-based. Components are the building blocks of any React application, and a single app usually consists of multiple components. A component might include one or more other components. Components can be styled with CSS. Listing 5.6 represents the App component, the top most component of the frontend application.

```
import React from 'react'

import Header from './components/pages/Header/Header'
import Sidebar from './components/pages/Sidebar/Sidebar'
import Routes from './components/routes/Routes'
import { AuthProvider } from './components/auth/AuthContext'

import menuItems from './components/routes/menuItems'

function App() {
    return (
        <AuthProvider>
            <div className='wrapper'>
                <Sidebar
                    menuItems={menuItems}
                ></Sidebar>
                <Header></Header>
                <div className='main'>
                    <Routes></Routes>
                </div>
            </div>
        </AuthProvider>
    )
}

export default App
```

Listing 5.6: App component

In order to enhance the interactivity with the tool and the quality of the information displayed, a few libraries were used.

React-datepicker (HackerOne 2020) is a simple and reusable datepicker component for React. The user can simply click on the datepicker, open up a calendar and select the desired date. It also comes with useful customisable options, such as defining the min and max date the user is able to pick and the format the date should be displayed in.

Moment.js (Moment.js 2020) is a lightweight JavaScript date library for parsing, validating, manipulating, and formatting dates. This library was used because the default Date object in JavaScript does not have enough functions to handle dates efficiently. It was especially useful to handle date formats, calculate the difference between two dates and add days to a date.

React-wordcloud (React Wordcloud 2020) is a simple React + D3 word cloud component with powerful features that uses the d3-cloud layout. This library was used to display the frequency of keywords for a given date interval. The size of the keyword in the word cloud is relative to its frequency. With many customisable options, all that is needed to create a word cloud is an array of word objects, in which the word object takes the following shape: { text: string, value: number}.

React-chartjs-2 is a React wrapper for Chart.js (Chart.js 2020), a JavaScript library for data visualisation. This library was used to create highly customisable bar, line and doughnut charts that display different kinds of information. To create a chart, all that is needed is an array of labels and an array of one or more corresponding datasets.

The following sections showcase the final user interface.

5.3.1 Praxis Market Drift sitemap

Figure 5.2 represents the hierarchical structure of the Praxis Market Drift website, composed by four main pages: dashboard, searches, offers and clusters. The clusters page can either display global clusters or periodic clusters.

Figure 5.2: Praxis Market Drift sitemap.

5.3.2 Dashboard

The dashboard is the home page for the application, the page the user sees right after logging in. The aim of this page is to present general and recent statistics about the Praxis platform. Starting from the top, the user can select whether he wants to see information regarding the last 30 days, the last 7 days or the last day. The next six blocks present information regarding the total number of searches made, the total number of offers submitted, the total number of active offers, the average number of results each search gets and the number

of students and providers that logged in to the platform. Even though these statistics are the result of simple database queries and not text mining techniques, the information is still insightful to the user. The last two blocks are composed of two bar charts that present data relative to keyword frequencies in searches and offers. Since the dashboard is supposed to keep it simple and not go in-depth, it was decided that the bar charts would only show the top 5 keywords. Figure 5.3 represents the dashboard page.

Figure 5.3: Dashboard page.

5.3.3 Searches and Offers

The searches and offers pages are similar and both go more in-depth when it comes to analysing the evolution of the market. For starters, the user can now select the exact date interval he wants to analyse through the datepickers at the top of the page. Whenever the user picks a new date, the word cloud is immediately updated with the words and their respective frequencies for the respective date interval. The word cloud presents the data in a way that the most used keywords have a bigger font size, and when hovering over a keyword, the word cloud highlights that keyword and displays a tooltip with its exact frequency value.

Taking it a step further, clicking on a keyword shows the evolution of that keyword in a line chart. The x axis represents all the dates between the selected date interval and the y axis the number of keyword occurrences for each specific date. The line chart can display multiple keywords at the same time, and clicking on a keyword label makes it so the keyword is not displayed in the chart. This offers the user great flexibility since he can easily make comparisons between datasets. To completely remove a keyword from the line chart, the user just needs to click on that keyword in the word cloud again. The user can also hover over the data points in the line chart to see the exact value for each date.

Combining the ability to view the most used keywords in a date interval with the ability to view when exactly they were used the most allows the user to conduct an insightful and precise analysis of the evolution of the Praxis market. Figure 5.4 represents the searches page.

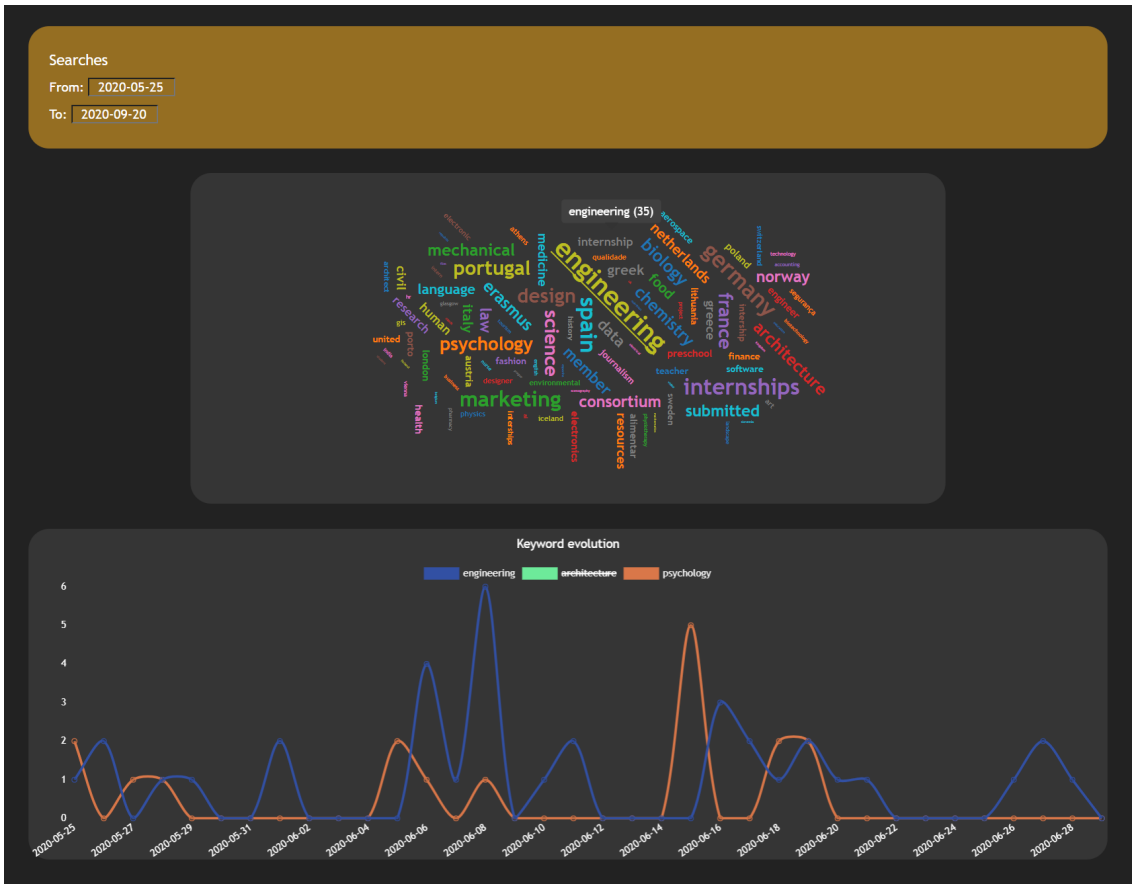

Figure 5.4: Searches page.

5.3.4 Clusters

The clusters page presents the results generated by the Carrot2 API through a doughnut chart. Each slice in the chart corresponds to a cluster, the bigger the slice the bigger the amount of offers contained in the cluster. Just like in the last pages, when a new date is selected, the doughnut chart is updated with the new clusters for the selected date interval. The user has the option to choose which algorithm the Carrot2 API should use to generate the clusters, and the doughnut chart updates when a new algorithm is selected. Hovering over a slice of the doughnut chart displays a tooltip with the cluster's label and the amount of offers contained in that cluster. Clicking on a slice displays a detailed list of the offers contained in that cluster, showing the id, title and link for each offer. The user can click on an offer to be redirected to its page on the official Praxis website. This functionality is available for both global clusters and periodic clusters. For the selected date interval, the

user can either view a global cluster, represented in Figure 5.5, or a set of periodic clusters, represented in Figure 5.6.

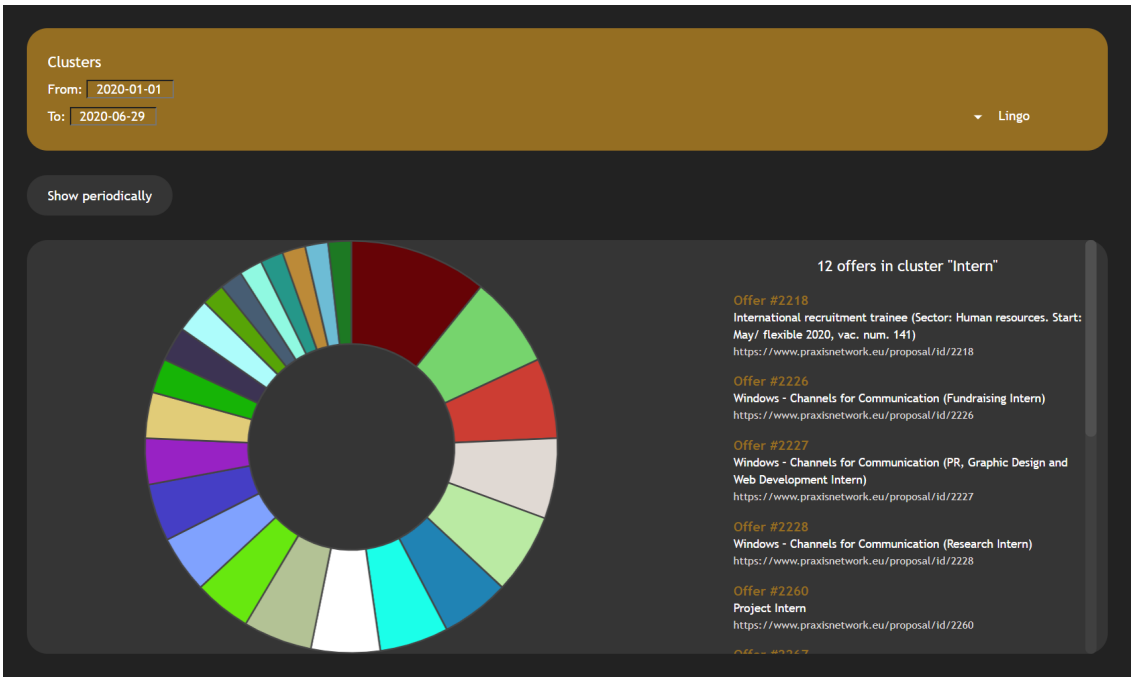

Figure 5.5: Clusters page - Global cluster.

Figure 5.6: Clusters page - Periodic clusters.

The date interval is divided into six periods, to which one cluster is generated for each period. Six seemed the ideal number between displaying too much and not enough information on the screen.

Since clustering groups offers based on similar characteristics, the user is able to know what kind of offers are being submitted the most through the Praxis platform for the selected date interval, and he can even conduct a deeper analysis by viewing the periodic clusters. Being able to choose between different algorithms also offers great flexibility since each algorithm generates different results.

Chapter 6

Evaluation

This chapter evaluates the final Praxis Market Drift solution using satisfaction surveys, where the main features of the solution are analysed. It also includes an evaluation of the clustering algorithms used in this thesis.

6.1 Evaluation methodology

To evaluate the developed solution, a satisfaction survey was carried out to a group of people in the Praxis network.

The primary goal of this survey is to collect opinions from people familiarised with the Praxis platform, whether they use it to search for internship offers or to submit internship offers, to know if the data displayed is good enough to understand the evolution of the Praxis' internships market. Therefore, the survey aims to portray diverse points of view when it comes to the various features offered by the Praxis Market Drift solution.

For the survey, a set of statements were created and classified according to the Likert scale, to measure the participants' level of agreement with each statement.

The Likert scale is a rating scale used to assess opinions, attitudes, or behaviors (Bhandari 2020). The survey used a five-point satisfaction scale, represented in Table 6.1.

Table 6.1: Likert Scale.

Level	Description
1	Strongly disagree
2	Disagree
3	Neutral
4	Agree
5	Strongly agree

The solution was also evaluated when it comes to the results of the clustering algorithms used to group offers based on similar characteristics. The main goal of evaluating the

clustering algorithms is to figure out which one presents the better results for the same dataset and how each algorithm reacts to different dataset sizes.

6.2 Survey results

The satisfaction survey carried out to the Praxis network counted with the participation of 12 people from various different countries, which had to answer a total of 7 statements. Out of the 12 people that answered the survey, 6 use Praxis to search for internship offers, 4 use Praxis to submit internship offers and 2 use Praxis for both. The participants' country and Praxis usage distributions are represented, respectively, in Figure 6.1 and Figure 6.2.

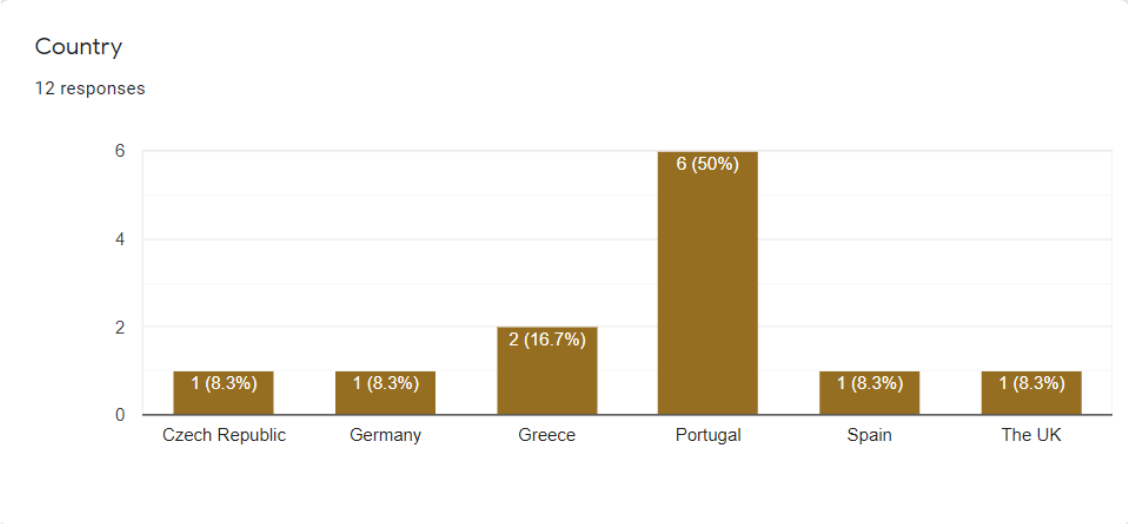

Figure 6.1: Participant country distribution.

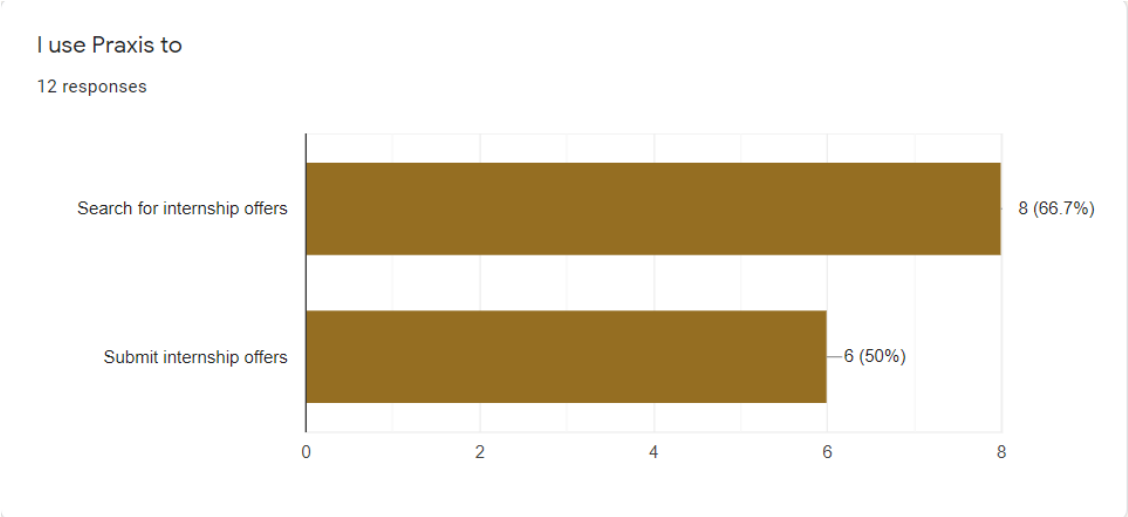

Figure 6.2: Participant Praxis usage distribution.

The participants had to answer the following statements:

- S1: The dashboard provides a simple and informative overview about the recent usage of the Praxis platform in regards to available internship offers and student searches.
- S2: The combination of using wordclouds to display keyword frequencies in user searches with the ability to view the exact number of occurrences for a specific date allows for an in-depth analysis of the evolution of users' interests over time.
- S3: Using clusters to group offers based on similar characteristics allows for an analysis of the evolution of the internship subjects being submitted through Praxis.
- S4: The ability to view clusters for different periods in a date interval enhances the analysis of the evolution of offers submitted through Praxis.
- S5: The possibility to use different clustering algorithms is helpful.
- S6: Praxis Market Drift is user friendly and simple to use without prior training.
- S7: Praxis Market Drift is a powerful tool to analyse and understand the internships market and its evolution over time.

According to the answers obtained, the survey results were, in general, quite positive. The answers to each statement in the survey can be visualised in Table 6.2. These answers can be graphically visualised in Appendix B.

Table 6.2: Survey answers.

	1	**2**	**3**	**4**	**5**
S1	0	0	1(8.3%)	4(33.3%)	7(58.3%)
S2	0	0	1(8.3%)	4(33.3%)	7(58.3%)
S3	0	0	0	2(16.7%)	10(83.3%)
S4	0	0	1(8.3%)	4(33.3%)	7(58.3%)
S5	0	0	0	5(41.7%)	7(58.3%)
S6	0	0	1(8.3%)	1(8.3%)	10(83.3%)
S7	0	0	1(8.3%)	3(25%)	8(66.7%)

In the first statement, 1 participant selected the level of agreement 3 (neutral), 4 participants selected the level of agreement 4 (agree) and 7 participants selected the level of agreement 5 (strongly agree). Therefore, the dashboard does a good job when it comes to conveying simple yet relevant information about the recent usage of the Praxis platform.

In the second statement, 1 participant selected the level of agreement 3 (neutral), 4 participants selected the level of agreement 4 (agree) and 7 participants selected the level of agreement 5 (strongly agree). So, the text mining techniques used to obtain meaningful information and the types of data visualisation chosen to display such information are appropriate to analyse the evolution of users' interests over time.

In the third statement, 2 participants selected the level of agreement 4 (agree) and 10 participants selected the level of agreement 5 (strongly agree). Thus, the overwhelmingly positive response to this statement supports the choice of using clustering techniques to analyse the evolution of the internship offers submitted.

In the fourth statement, 1 participant selected the level of agreement 3 (neutral), 4 participants selected the level of agreement 4 (agree) and 7 participants selected the level of agreement 5 (strongly agree). Therefore, viewing clusters for different periods in a date interval is a good complement to further analyse the evolution of the internship offers submitted.

In the fifth statement, 5 participants selected the level of agreement 4 (agree) and 7 participants selected the level of agreement 5 (strongly agree). So, being able to choose between 3 algorithms that provide different results is advantageous for the user conducting the analysis.

In the sixth statement, 1 participant selected the level of agreement 3 (neutral), 1 participant selected the level of agreement 4 (agree) and 10 participants selected the level of agreement 5 (strongly agree). Thus, the majority of the participants agree that Praxis Market Drift is an intuitive and simple to use tool.

In the seventh and final statement, 1 participant selected the level of agreement 3 (neutral), 3 participants selected the level of agreement 4 (agree) and 8 participants selected the level of agreement 5 (strongly agree). The answers to this statement represent the overall rating for the Praxis Market Drift tool. According to the results, the developed solution accomplishes the defined objectives.

6.3 Clustering algorithms evaluation

Even though the solution provides the choice of using any of the three algorithms provided by the Carrot2 DCS, Lingo, STC and K-Means, it is still important to know how each algorithm interacts with different dataset sizes. Each algorithm was tested for two different criteria: cluster diversity and processing time. The cluster diversity criteria represents how descriptive the clusters created are, how well the algorithm organises documents into different categories, and the processing time criteria represents the time it takes to generate the clusters. Three different datasets with different sizes were used for the tests, using data from the Praxis database: the first dataset holds a total of 71 documents, the second dataset holds a total of 934 documents and the third holds a total of 2183 documents. The results for each algorithm are present in Table 6.3.

Table 6.3: Test results for Lingo, STC and K-Means algorithms.

	Number of documents	**Cluster diversity**	**Processing time (ms)**
Lingo	71	High	60
STC	71	High	40
K-Means	71	Medium	95
Lingo	934	High	600
STC	934	Medium	90
K-Means	934	Low	450
Lingo	2183	High	2900
STC	2183	Medium	160
K-Means	2183	Low	1000

In terms of scalability, the STC algorithm shows impressive results. The algorithm processing time is extremely quick independently of the number of documents. On the other hand, Lingo and STC get considerably slower as the number of documents increases. Lingo, especially, starts taking longer when the number of documents passes the 1000 mark.

In terms of cluster diversity, Lingo shows by far the best results. As the number of documents increases, the algorithm remains consistent when grouping documents into intuitive and diverse clusters. STC and K-Means show good results for a low number of documents, but as the number of documents goes up, the quality of the clusters decreases. This can be explained by the fact that Lingo, in general, creates more clusters with more descriptive labels.

Additionally, Lingo and STC have the advantage of being able to allocate a document to more than one cluster, while with K-Means a document can belong to only one cluster.

In conclusion, the Lingo algorithm seems overall the best choice to fulfil the needs of the Praxis Market Drift solution. While STC and K-Means show better performance results, Lingo generates more insightful clusters, which is more important for the overall goal of the application.

Chapter 7

Conclusion

This chapter reflects on all the work that went into developing the Praxis Market Drift solution. It goes over the achieved objectives, underlines the limitations faced during the development of the solution and presents future work suggestions.

7.1 Achieved objectives

This dissertation had as main objective the development of a flexible tool to analyse and understand the Praxis internships market and its evolution over time. To do so, the tool needed to be able to display all the necessary information to the Praxis administrators, regarding user keyword searches and offers subjects being submitted through Praxis.

The development of this project showcased the importance of text mining and how it can be utilised to extract insightful information from unstructured text data. It explored various text mining techniques and how each could be used to achieve the defined goals, from preprocessing techniques to counting the frequency of words in documents to grouping documents that share similar characteristics into clusters.

It exposed the importance of data visualisation as a complement to text mining techniques, and how modern web frameworks can be used to enhance the data displayed to the user. It also mentions that the choice of the type of visualisation to use must take into consideration the intended purpose of the data being displayed.

The final Praxis Market Drift solution was evaluated through a satisfaction survey, sent out to a group of people in the Praxis network. The responses to the survey were extremely positive, which means the objectives defined for the application were achieved with success.

All the work put into this dissertation was as relevant to the author as it was to Praxis. The author, previously unfamiliar with text mining, got to explore this interesting domain and learn plenty of new technologies. Praxis got a powerful tool to analyse and understand the internships market and its evolution over time.

7.2 Limitations

The data provided for the development of this project was dated June 29, 2020, which means it did not include the most recent activity in Praxis. Even though this was not a big limitation, it would have still been interesting to interact with real-time data to obtain the most recent information.

7.3 Future work

The current stopword list only includes the most common words in the Portuguese and English languages. In the future it would be interesting to create a custom stopword list for the specific Praxis domain. The ideal scenario would be to let the Praxis administrator set up a list of stopwords himself in the Praxis Market Drift tool, so that irrelevant words, that do not add anything to the analysis of the evolution of the internships market, would not appear in the results.

As mentioned in Section 4.3, it was decided to adopt the first architecture proposal to build the system, keeping in mind that it should easily be upgradable to the second architecture. As the amount of data grows in the Praxis database, it is recommended to upgrade the system to adopt the second architecture.

Bibliography

Allahyari, Mehdi et al. (July 2017). "A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques". In: *KDD Bigdas*. url: https://arxiv.org/pdf/1707.02919.pdf.

Amendo (Nov. 2018). "Business Model Canvas". In: *Medium*. url: https://medium.com/amendo_s/business-model-canvas-ecb11d6fe4 (visited on 2020-02-03).

Analytiks (June 2020). "Why Data Visualization Is Important". In: *Analytiks*. url: https://analytiks.co/importance-of-data-visualization/ (visited on 2020-09-26).

Bhandari, Pritha (July 2020). "Designing and analyzing Likert scales". In: *Scribbr*. url: https://www.scribbr.com/methodology/likert-scale (visited on 2020-10-07).

Bird, Steven, Ewan Klein, and Edward Loper (2009). *Natural Language Processing with Python*. 1st ed. O'Reilly Media Inc. url: https://www.nltk.org/book/.

Boehmke, Bradley (Jan. 2017). "Text Mining: Term vs. Document Frequency". In: url: http://uc-r.github.io/tf-idf_analysis (visited on 2020-09-25).

Boostlabs (Nov. 2014). "Word Clouds & the Value of Simple Visualizations". In: *Boostlabs*. url: https://boostlabs.com/blog/what-are-word-clouds-value-simple-visualizations/ (visited on 2020-09-26).

Bostock, Michael, Vadim Ogievetsky, and Jeffrey Heer (Oct. 2011). "D3: Data-Driven Documents". In: url: http://idl.cs.washington.edu/files/2011-D3-InfoVis.pdf.

Carrot Search (2020). *Hello, Carrot2!* url: https://carrot2.github.io/release/4.0.0/doc/ (visited on 2020-09-28).

Carrot2 (2020). *Carrot2 search results clustering engine*. url: https://search.carrot2.org (visited on 2020-09-26).

Chang, Jiakang et al. (2018). "Text mining 101". In: *FOSTER*. url: https://www.fosteropenscience.eu/content/text-mining-101 (visited on 2020-01-23).

Chart.js (2020). *Chart.js | Open source HTML5 Charts for your website*. url: https://www.chartjs.org (visited on 2020-09-14).

D3 (2020). *D3.js - Data-Driven Documents*. url: https://d3js.org/ (visited on 2020-09-26).

Docker (2020a). *Docker: Empowering App Development for Developers*. url: https://www.docker.com (visited on 2020-09-12).

– (2020b). *Overview of Docker Compose*. url: https://docs.docker.com/compose (visited on 2020-09-12).

Enterprise Big Data Framework (Jan. 2019). "Data Types: Structured vs. Unstructured Data". In: *Enterprise Big Data Framework*. url: https://www.bigdataframework.org/data-types-structured-vs-unstructured-data/ (visited on 2020-01-15).

Expert System (Sept. 2016). "The value of text mining for unstructured information". In: *Expert System*. url: https://expertsystem.com/value-text-mining-unstructured-information/ (visited on 2020-01-16).

– (May 2020). "Natural language processing and text mining". In: *Expert System*. url: https://expertsystem.com/natural-language-processing-and-text-mining (visited on 2020-10-12).

Feinerer, Ingo (Dec. 2019). "Introduction to the tm Package". In: url: https://cran.r-project.org/web/packages/tm/vignettes/tm.pdf (visited on 2020-03-05).

Flask (2020). *Welcome to Flask — Flask Documentation*. url: https://flask.palletsprojects.com (visited on 2020-09-13).

Flask-RESTful (2020). *Flask-RESTful — Flask-RESTful documentation*. url: https://flask-restful.readthedocs.io (visited on 2020-09-13).

Florén, Henrik and Johan Frishammar (Sept. 2010). "Achieving Success in the Fuzzy Front End Phase of Innovation". In: *Innovation Management*. url: https://innovationmanagement.se/2010/10/20/achieving-success-in-the-fuzzy-front-end-phase-of-innovation/ (visited on 2020-10-10).

Ganesan, Kavita (Apr. 2019). "All you need to know about text preprocessing for NLP and Machine Learning". In: *KDnuggets*. url: https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html (visited on 2020-09-24).

Google (2020). *Google Trends*. url: https://trends.google.pt (visited on 2020-09-26).

HackerOne (2020). *React Datepicker*. url: https://reactdatepicker.com (visited on 2020-09-14).

Haksever, Cengiz, Radha Chaganti, and Ronald G. Cook (Feb. 2004). "A Model of Value Creation: Strategic View". In: *Journal of Business Ethics*. url: https://www.researchgate.net/publication/226872391_A_Model_of_Value_Creation_Strategic_View.

Hassan, Almoatazbillah (June 2012). "The Value Proposition Concept in Marketing: How Customers Perceive the Value Delivered by Firms". In: url: https://www.researchgate.net/publication/268370501_The_Value_Proposition_Concept_in_Marketing_How_Customers_Perceive_the_Value_Delivered_by_Firms_-_A_Study_of_Customer_Perspectives_on_Supermarkets_in_Southampton_in_the_United_Kingdom.

IDC (2020). *Global DataSphere*. url: https://www.idc.com/getdoc.jsp?containerId=IDC_P3835 (visited on 2020-01-15).

import.io (Oct. 2019). "What is Data Visualization and Why Is It Important?" In: *import.io*. url: https://www.import.io/post/what-is-data-visualization/ (visited on 2020-09-26).

Jinha, Arif E. (July 2010). "Article 50 million: an estimate of the number of scholarly articles in existence". In: *Learned Publishing* 23, pp. 258–263. url: https://doi.org/10.1087/20100308.

Jivani, Anjali Ganesh (Nov. 2011). "A Comparative Study of Stemming Algorithms". In: *Int. J. Comp. Tech. Appl.* url: https://kenbenoit.net/assets/courses/tcd2014qta/readings/Jivani_ijcta2011020632.pdf.

Koen, Peter A. et al. (2002). *Fuzzy Front End:Effective Methods, Tools, and Techniques*. The PDMA Toolbook for new product development. url: https://www.academia.edu/33974079/1_Fuzzy_Front_End_Effective_Methods_Tools_and_Techniques.

Li, Mei-Lien and Robert D. Green (2011). "A mediating influence on customer loyalty: The role of perceived Value". In: *Journal of Management and Marketing Research*. url: https://pdfs.semanticscholar.org/1e75/32029f8c3e5ad6561bbbb0f41e56f1b9e9e8.pdf.

Mayo, Matthew (Dec. 2017). "A General Approach to Preprocessing Text Data". In: *KDnuggets*. url: https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html (visited on 2020-09-23).

Mohler, Tim (Sept. 2019). "The 7 Basic Functions of Text Analytics". In: *Lexalytics*. url: https://www.lexalytics.com/lexablog/text-analytics-functions-explained (visited on 2020-09-23).

Moment.js (2020). *Moment.js*. url: https://momentjs.com (visited on 2020-09-14).

MonkeyLearn (2020). "Text Mining: The Beginner's Guide". In: *MonkeyLearn*. url: https://monkeylearn.com/text-mining/ (visited on 2020-01-20).

MySQL (2020). *MySQL Connector/Python Developer Guide*. url: https://dev.mysql.com/doc/connector-python/en (visited on 2020-09-13).

Osinski, Stanislaw (2003). *AN ALGORITHM FOR CLUSTERING OF WEB SEARCH RESULTS*. url: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.5832&rep=rep1&type=pdf.

Osinski, Stanislaw and Dawid Weiss (2004). "Conceptual Clustering Using Lingo Algorithm: Evaluation on Open Directory Project Data". In: url: https://core.ac.uk/download/pdf/191307876.pdf.

Padmanabhuni, Sri Shilpa and Hima Bindu (Oct. 2013). "Search Results Clustering: Comparison of Lingo and K-Means". In: url: https://www.academia.edu/5107650/Comparison_of_Lingo_and_K_Means.

Passage Technology (2020). "What Is The Analytic Hierarchy Process (AHP)?" In: *Passage Technology* (). url: https://www.passagetechnology.com/what-is-the-analytic-hierarchy-process (visited on 2020-09-29).

Praxis Network (2020). *Student internships and challenges in PRAXIS*. url: https://www.praxisnetwork.eu (visited on 2020-01-16).

React (2020). *React – A JavaScript library for building user interfaces*. url: https://reactjs.org (visited on 2020-09-14).

React Wordcloud (2020). *React Wordcloud*. url: https://react-wordcloud.netlify.app (visited on 2020-09-14).

Reinsel, David, John Gantz, and John Rydning (May 2020). "The Digitization of the World". In: *Seagate*. url: https://www.seagate.com/files/www-content/our-story/trends/files/dataage-idc-report-final.pdf.

Relocate Magazine (2019). *The rise of global student mobility*. url: https://www.relocatemagazine.com/articles/education-schools-international-guide-2017-the-rise-of-global-student-mobility (visited on 2020-01-17).

Reports and Data (2020). *Data Mining Tools Market*. url: https://www.reportsanddata.com/report-detail/data-mining-tools-market (visited on 2020-02-01).

SAVE (2020). "About Value Engineering". In: *SAVE* (). url: https://www.value-eng.org/page/AboutVE (visited on 2020-02-01).

scikit-learn (2020). *scikit-learn: Machine Learning in Python*. url: https://scikit-learn.org (visited on 2020-09-28).

Seif, George (Feb. 2018). "The 5 Clustering Algorithms Data Scientists Need to Know". In: *Towardsdatascience*. url: https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68 (visited on 2020-02-03).

Silge, Julia and David Robinson (Mar. 2020). "Text Mining with R: Analyzing word and document frequency: tf-idf". In: url: https://www.tidytextmining.com/tfidf.html (visited on 2020-09-25).

spaCy (2020). *spaCy 101: Everything you need to know*. url: https://spacy.io/usage/spacy-101#_title (visited on 2020-09-27).

The R Foundation (2020). *What is R?* url: https://www.r-project.org/about.html (visited on 2020-03-05).

Ware, Mark and Michael Mabe (Mar. 2015). "An overview of scientific and scholarly journal publishing". In: *The STM report*. url: https://www.stm-assoc.org/2015_02_20_STM_Report_2015.pdf.

Woodall, Tony (Jan. 2003). "Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis". In: url: https://www.researchgate.net/publication/228576532_Conceptualising_'Value_for_the_Customer'_An_Attributional_Structural_and_Dispositional_Analysis.

WordStream (2020). *Free Keyword Tool*. url: https://www.wordstream.com/keywords (visited on 2020-09-26).

Appendix A

Full API specification

Table A.1: Searches count request.

HTTP method	GET
Endpoint	/api/searches/count/<int:lastDays>
Description	Returns the number of user searches in the last specified days Possible values for lastDays: 30, 7, 1

Response example	{
	"nSearches": 3481
	}

Table A.2: Offers count request.

HTTP method	GET
Endpoint	/api/offers/count/<int:lastDays>
Description	Returns the number of new offers submitted in the last specified days Possible values for lastDays: 30, 7, 1

Response example	{
	"nOffers": 14
	}

Table A.3: Active offers count request.

HTTP method	GET
Endpoint	/api/offers/count/active/<int:lastDays>
Description	Returns the number of active offers for the last specified days Possible values for lastDays: 30, 7, 1

Response example	{
	"nOffers": 149
	}

Table A.4: Average results per search request.

HTTP method	GET
Endpoint	/api/searches/avgresultspersearch/<int:lastDays>
Description	Returns the number of average results per search for the last specified days Possible values for lastDays: 30, 7, 1

Response example	{
	"avgResultsPerSearch": 9.76
	}

Table A.5: Student logins request.

HTTP method	GET
Endpoint	/api/users/students/loginCount/<int:lastDays>
Description	Returns the number of student logins for the last specified days Possible values for lastDays: 30, 7, 1

Response example	{
	"studentLogins": 49
	}

Table A.6: Provider logins request.

HTTP method	GET
Endpoint	/api/users/providers/loginCount/<int:lastDays>
Description	Returns the number of provider logins for the last specified days Possible values for lastDays: 30, 7, 1

Response example	{
	"providerLogins": 6
	}

Table A.7: Searches keyword frequency request.

HTTP method	GET
Endpoint	/api/searches/keywordFrequency/<string:fromDate>/<string:toDate>/<int:keywordNumber> /api/searches/keywordFrequency/<int:lastDays>/<int:keywordNumber>
Description	Returns searches keyword frequencies for the last specified days or for a date interval fromDate: start date (yyyy-mm-dd) toDate: end date (yyyy-mm-dd) keywordNumber: max number of keywords to return Possible values for lastDays: 30, 7, 1

Response example	[
	{
	"text": "engineering",
	"value": 349
	}
	]

Table A.8: Offers keyword frequency request.

HTTP method	GET
Endpoint	/api/offers/keywordFrequency/<string:fromDate>/<string:toDate>/<int:keywordNumber> /api/offers/keywordFrequency/<int:lastDays>/<int:keywordNumber>
Description	Returns offers keyword frequencies for the last specified days or for a date interval fromDate: start date (yyyy-mm-dd) toDate: end date (yyyy-mm-dd) keywordNumber: max number of keywords to return Possible values for lastDays: 30, 7, 1

Response example	[
	{
	"text": "internship",
	"value": 34
	}
	]

Table A.9: Searches keyword evolution request.

HTTP method	GET
Endpoint	/api/searches/keywordEvolution/<string:fromDate>/<string :toDate>/<string:keyword>
Description	Returns the number of occurrences of a keyword in searches for each date in the date interval fromDate: start date (yyyy-mm-dd) toDate: end date (yyyy-mm-dd) keyword: the name of the keyword

Response example	[
	{
	"keywordCount": 2,
	"date": "2020-01-01",
	}
	]

Table A.10: Offers keyword evolution request.

HTTP method	GET
Endpoint	/api/offers/keywordEvolution/<string:fromDate>/<string:t oDate>/<string:keyword>
Description	Returns the number of occurrences of a keyword in offers for each date in the date interval fromDate: start date (yyyy-mm-dd) toDate: end date (yyyy-mm-dd) keyword: the name of the keyword

Response example	[
	{
	"keywordCount": 2,
	"date": "2020-01-01",
	}
	]

Table A.11: Clusters request.

HTTP method	GET
Endpoint	/api/clusters/<string:fromDate>/<string:toDate>/<string:algorithm>
Description	Returns clusters for a set of offers in the date interval fromDate: start date (yyyy-mm-dd) toDate: end date (yyyy-mm-dd) Possible values for algorithm: lingo, stc, bkmeans

Response example	{
	"clusters" : [
	{
	"labels" : [
	"Graphic Design"
	],
	"documents" : [
	{
	"id": 2263,
	"title": "Graphic Designer / Researcher"
	},
	],
	"clusters" : [ ],
	"score" : 34.56174212000691
	}
	]
	}

Appendix B

Satisfaction survey answers

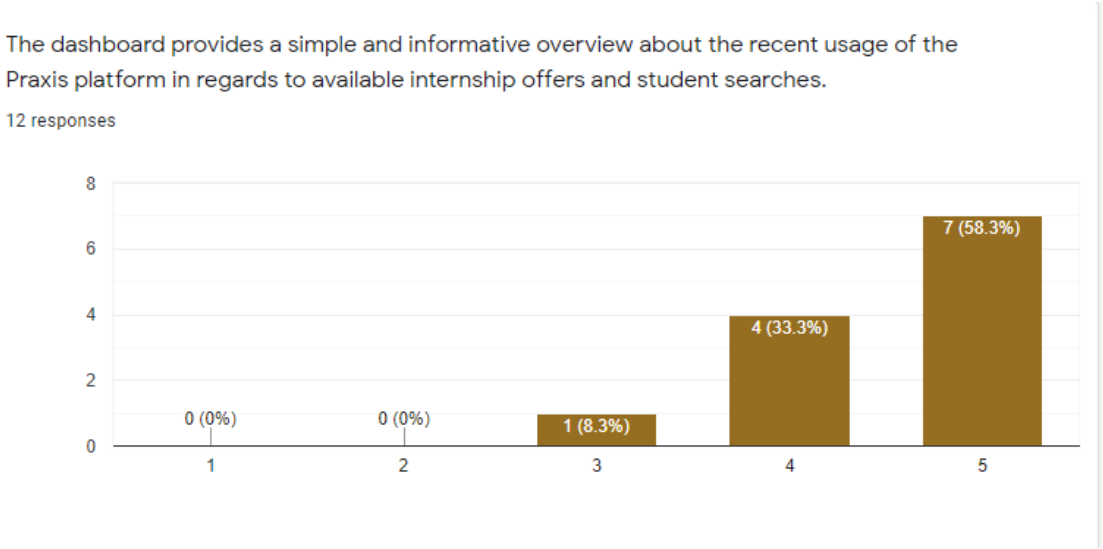

Figure B.1: Statement 1 answers.

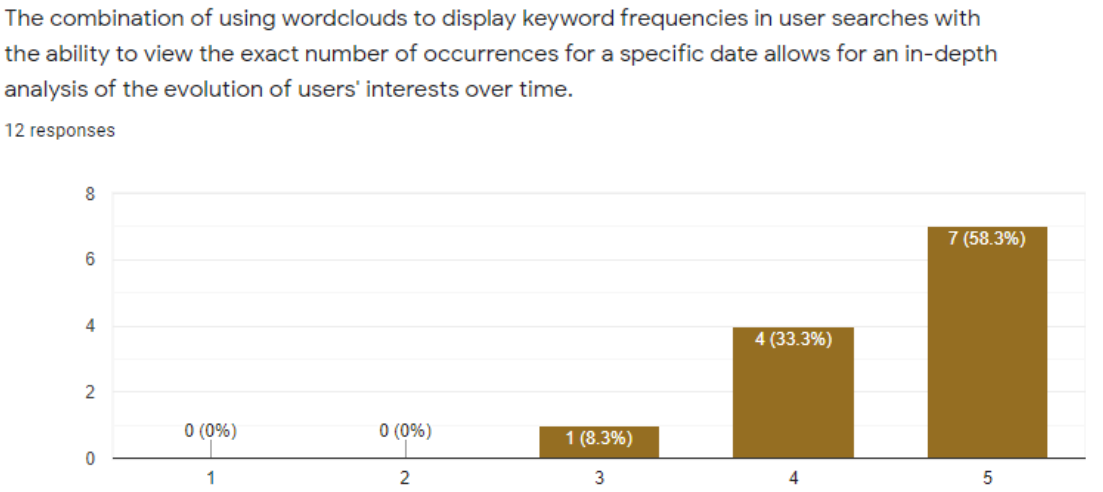

Figure B.2: Statement 2 answers.

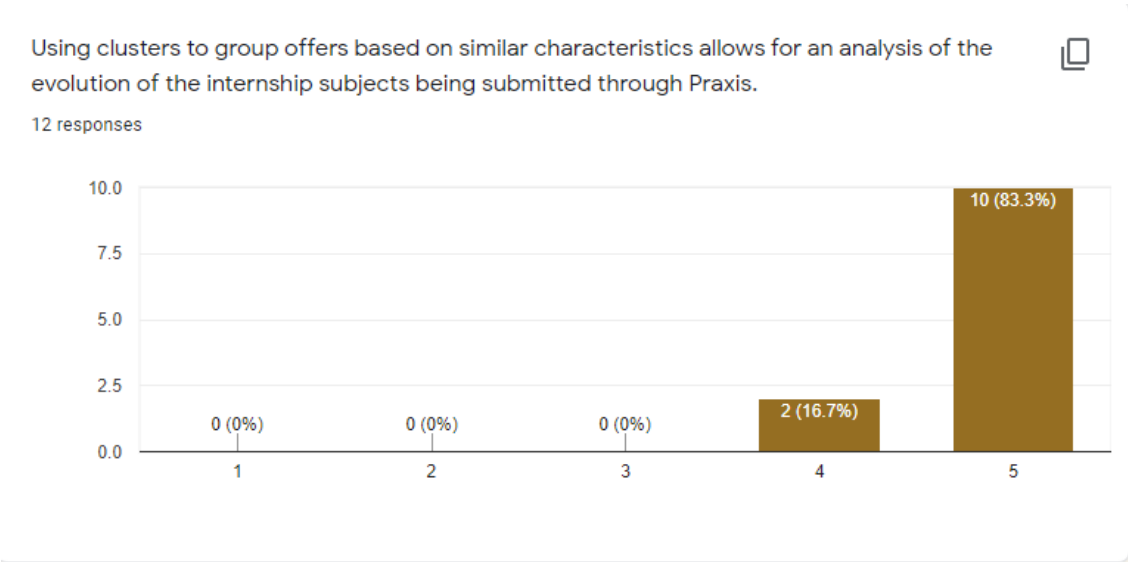

Figure B.3: Statement 3 answers.

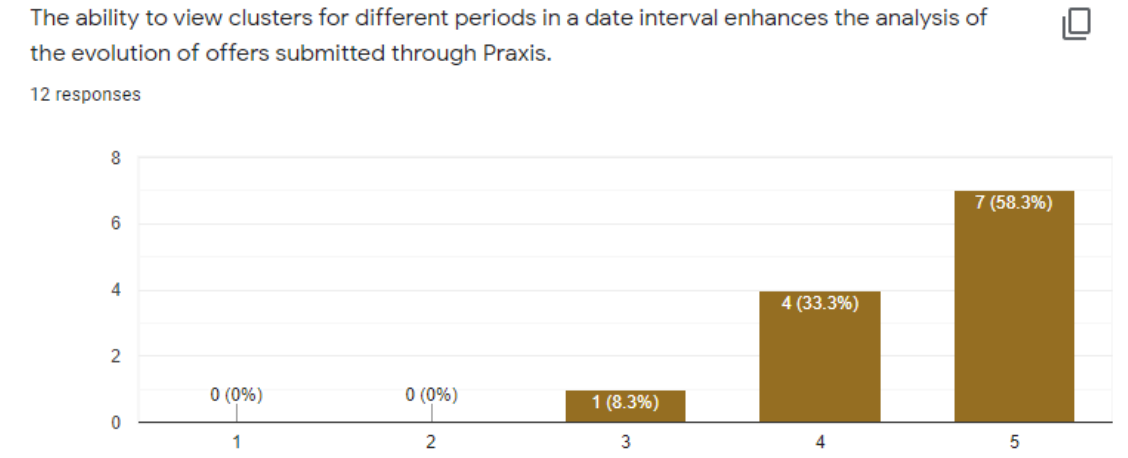

Figure B.4: Statement 4 answers.

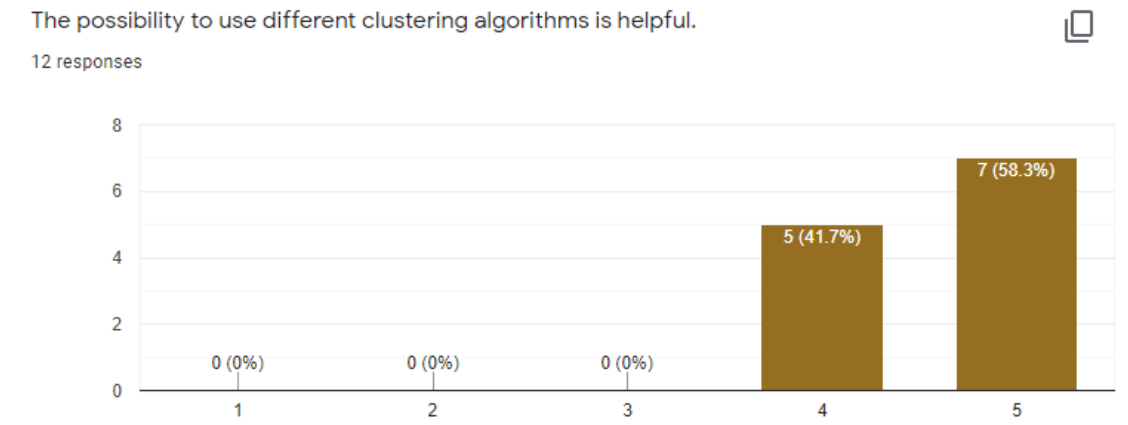

Figure B.5: Statement 5 answers.

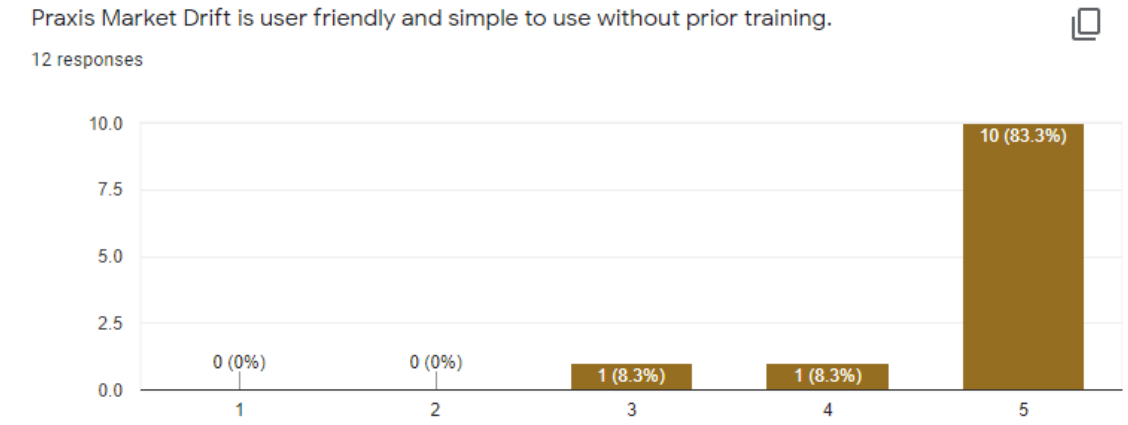

Figure B.6: Statement 6 answers.

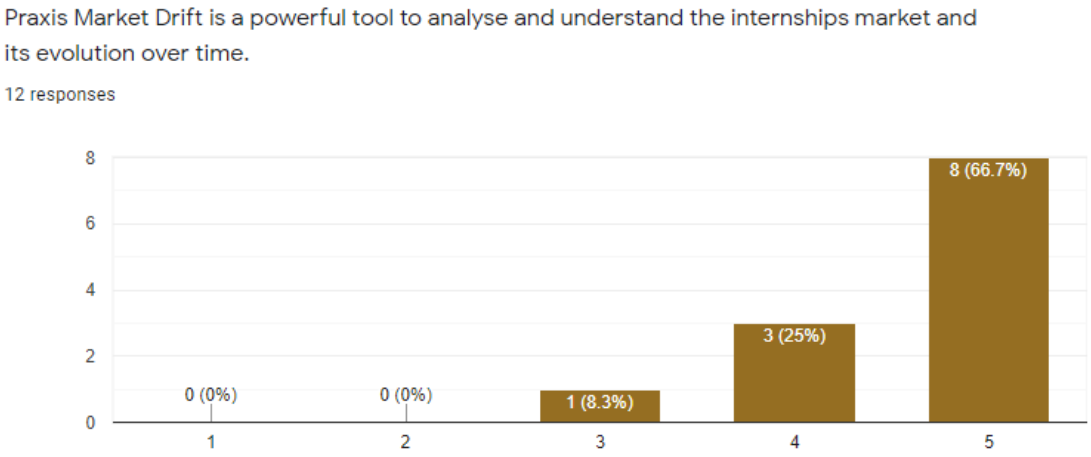

Figure B.7: Statement 7 answers.