Pedro Rafael
Gonçalves Miguel

Extração de Informação Aplicada a Comentários da Área do Turismo

Information Extraction Applied to Tourism Electronic Word-of-Mouth (eWOM)

Pedro Rafael
Gonçalves Miguel

Information Extraction Applied to Tourism Electronic Word-of-Mouth (eWOM)

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor António Joaquim da Silva Teixeira, Professor Associado com Agregação do Departamento de Eletrónica Telecomunicações e Informática da Universidade de Aveiro e do Doutor Mário Jorge Rodrigues, Professor Adjunto da Escola Superior de Tecnologia e Gestão de Águeda da Universidade de Aveiro.

o júri / the jury

presidente / president	**Professora Doutora Pétia Georgieva Georgieva** Professora Associada da Universidade de Aveiro
vogais / examiners committee	**Professora Doutora Liliana da Silva Ferreira** Professora Catedrática Convidada da Faculdade de Engenharia da Universidade do Porto
	Professor Doutor António Joaquim da Silva Teixeira Professor Associado com Agregação da Universidade de Aveiro (Orientador)

acknowledgements

Thanks to my thesis advisor, Professor António Joaquim da Silva Teixeira, for your support, patience, guidance, and perseverance. I really appreciated that you accepted me for this thesis, and even after a significant interruption in the development, you still took me in and guided me. Finally, thank you for motivating me to finish this thesis. Without the professor help, this thesis wouldn't be the same.

Thanks to my thesis co-advisor, Professor Mário Jorge Ferreira Rodrigues, for your support, for your pieces of advice and for all the help you gave me. Thank you for always being present and ready to help me every time I needed it and for the tedious task of correcting my English.

Finally, thanks to my family for always supporting me during this event.

Palavras-chave NLP, parser, análise de sentimentos, tokenizer, sentence splitter, lemmatization, NER, crawler, ontology, POS, hotelaria, extração de informação.

Resumo

Motivação:A principal motivação por trás desta tese foi mostrar que é possível escrever um programa para NLP usando a língua portuguesa.

Objectivo(s):O principal objectivo desta tese foi extrair informação hotel dos comentários feitos a hotéis usando NLP.

Método: Foi criado um pipeline de NLP para extrair informação útil. Depois foi usado análise de sentimentos para caracterizar essa informação.

Resultados: Depois de todos os comentários serem processados foi possível descobrir o que as pessoas gostam ou desgostam sobre um hotel.

Conclusions: As duas principais conclusões foram que era possível fazer NLP em português e que era possível extrair informação útil de milhar de comentários.

Keywords NLP, parser, sentiment analysis, tokenizer, sentence splitter, lemmatization, NER, crawler, ontology, POS, hotel industry, information extraction.

Abstract

Motivation: The primary motivation of this dissertation was to show that it is possible to construct an NLP solution for the Portuguese language capable of helping in the hotel industry.

Objective(s): The main objective of this dissertation was to extract useful information from hotel commentaries using NLP.

Method: An NLP pipeline was created to extract useful information, and then sentimental analyse was used to characterise that information.

Results: After processing all the commentaries of a hotel was possible to extract what people like or dislike about it.

Conclusions: The two main conclusions were that is possible to create a Portuguese NLP pipeline for the hotel industry, and that is possible to extract useful information from thousands of commentaries.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Information is a compelling and essential resource, and with the appearance of the Internet, information has become more accessible for all people. Nowadays, a search engine can return millions of results, and each of those results can have more than a thousand words, but having access to large amounts of data is not always that good. For example, in a study performed 2019 (centred around the Google search engine), the first three results had 75.1% of all the clicks, and results on the second page had less than 1% of all the clicks[1].

However, can we be sure that we got all the relevant information on our first webpage? If not, how many pages should we browser through? Some companies created sites and/or services that gather information from various sources to fix this problem. For example, the website "kuantokusta.pt" lets people find the price of a product across several retailers.

The main focus of the dissertation is the hotel industry. Today we already have sites like Booking and Trivago that already agglomerate information about the hotels. Some even try to gather commentaries from other websites to enlarge the amount of information they have. However, that also brings a new problem because the amount of data is just too large to be helpful. For example, a single hotel can have hundreds or even thousands of comments (for example, the hotel "Bahia De Santander" has 1.464 reviews on the Trivago website). Even with numeric ratings is difficult to do a search when the decision is tied to specific tastes.

For example, we cannot search for a hotel with more positive reviews about the bed or the breakfast. The main problem is that we have to much important information that is not organized. For example, if we want to compare the pool from a hotel with four more, we will need to read hundreds or thousands of comments just to compare the pool. Do users have that time?

To help solve these problems, we decided to create a solution that gathers comments about hotels and transforms all the opinions into information ready to be consumed. Want to see the best bed in Aveiro? A simple search can show it. Want to see all the comments

that were about the breakfasts at a specific hotel? A simple search can show it. Resolving this problem will be the primary motivation of this dissertation.

1.2 The problems

As seen before in the motivation, there are already many sites to get information about hotels. Still, just a few of that information can be considered knowledge (we can define knowledge as the relevant information we use to make decisions or draw conclusions).

The primary forms of knowledge displayed on most websites are the hotel location, room prices, and rating. Still, these websites also have a significant amount of raw data, which is the commentaries. The typical user can read them and gather a good amount of knowledge about them, but it will take them a long time, and most users will not do that.

So what kind of information is the user losing if he does not read the comments? Furthermore, is it worth it to read the comments if he already has access to the hotel ratting? Usually, the hotel ratting can already be a good way of knowing if a hotel is good or not, but it is not absolute. Imagine a hotel with a good rating but had a change of ownership in the last two months. The user will not know if the hotel is still fine unless he reads the recent comments. Now consider that an essential feature for the user is the food that the hotel serves. He can try to use the ratting to estimate if the food is good or not, but to know it, he would have to read all the comments to see what people have said about it. Now imagine that the user is indecisive about several hotels? Is he going to read thousands of opinions to make his decision?

There is much raw data that could be very useful if it was transformed into something easy to access and simple to read. This type of information could even be helpful to the hotel manager to see what they could improve or maintain to raise their customers' happiness.

1.3 Objectives

This dissertation has several objectives:

- The first and most important is to prove that creating and developing an NLP pipeline that can extract information using the Portuguese language is possible.
- The second is creating an algorithm that can gather information from source such as Tripadvisor or Booking.
- The third is to use the pipeline to extract all the vital information, analyse it to see what is good and what is not and then store it.
- For last is creating a platform capable of showing this information to the public.

1.4 Dissertation structure

The Dissertations is structured as follows:

Chapter 1 – In this first chapter, the motivation for the theme of this dissertation is explained, then the problems that currently exist are described, and the way we intend to resolve them is also detailed.

Chapter 2 – This chapter is introduced what NLP is and how it changed over time. It also shows what IE and its main uses are, how it changed from the beginning until now, its generic pipeline and a short description of its blocks. After this, we also talk about one of the most important blocks, NER (we can have the same input and have different NER systems depending on the information we want to extract. NER is what permits us to differentiate what is valuable information and what is not). We also detail the NER history, main uses and the different techniques that have been developed until now.

Chapter 3 – This chapter introduces the primary personas/actors that can interact with our system (the hotel manager and the hotel client), their goals, frustration and scenarios, and the requirements derived by those scenarios. We also have a system overview, an explanation of each block of the pipeline and its implementation.

Chapter 4 – This chapter shows the main results that have been obtained after implementing the solution proposed in this dissertation. It also contains statistics and how our results were used in other projects like the data2tex and the voluntage4seniors.

Chapter 5 – This chapter summarises what has been achieved in this dissertation, the main conclusions and what can be done in the future to enrich this project.

Chapter 2

Background and Related Work

This chapter will introduce the main techniques that we use through our dissertation, how they evolve through time, their importance, and the primary way they are implemented.

2.1 Natural Language Processing

Natural language processing(NLP) is responsible for transforming natural language into data a computer can understand, use, and store. This process should not be undermined by the type of language used (English, Portuguese ...), the size of the data or the category of the data (spoken language, text, keyboard input) [2].

The NLP is a versatile and robust field, and it can be used to translate text from one language to another [3], to extract information from raw text [4], to create chatbot and assistants [5], speech recognition [6], sentiment analyses and others [7].

2.1.1 NLP history

We can divide the NLP history into three important generations, the symbolic NLP (rule base NLP), the statistical NLP and the deep learning NLP (that can later be divided into neural NLP and transformers NLP) [8].

Symbolic NLP – The symbolic NLP or rule-based NLP was the first approach used to translate human language to computer language, and it was based on a set of manually created rules (typically by linguistics). These rules were later programmed into computers so they could now process raw data. One of the first uses of NLP was in 1954 by the Georgetown experiment that translated a series of Russian sentences into English [9]

Statistical NLP – The statistical NLP or machine learning NLP was developed to reduced the need of human annotation to a minimum.

Deep learning NLP– The objective of deep learning in NLP is to exploit computer capabilities to create and develop features and representations that can later be used to interpret raw data [10]. Today two of the most important techniques are:

Neural NLP– Neural networks, as the name implies, are a series of neurons or nodes (dispersed in a network), and each of these nodes is receiving input and supplying output [11]. This approach comes to replace the linear models used by machine learning with a non-linear neural-network model [12].

Transformers NLP– The transformers approach came to solve one of the problems of the previous techniques that is the unidirectionality and the limited choices of architectures that can be used during training [8]. The first major system created was the BERT (Bidirectional Encoder Representations from Transformers) [13]. Other important implementation are RoBERTa [14], ALBERT [15] and XLNET [16].

2.2 Information Extraction

The main goal of information extraction is to get structured information from the unstructured or semi-structured text(usually built by humans). Structured information is a critical asset that permits the system to "understand" the text, store it, transmit it, or display it. Basically, the IE process is responsible for identifying semantically defined entities and the relationships between them [17].

The importance of the IE application shines in fields like, for example, medicine, where medical practices generate a large amount of information about the patients. Unfortunately, most of that information is still recorded as unstructured text. With the help of IE applications, this information could be used to improve clinical routine care, support clinical research, and advance the personalization of medicine [18].

To reduce the manual effort required, a new paradigm [19] called Open IE was introduced. Usually, IE applications are used to get precise information relevant to a determinate topic. For example, in the medical field is essential to extract names (medicines, doctor, patient, diseases ...), date and locations. Apart from the raw text, we also need a well-strutted corpus(usually created by linguistics). The principal focus of Open IE is to extract all relations found in a text instead of only extracting the relation derived by the corpus. To do this, Open IE had to overcome three significant challenges [20]:

- Automation, the IE applications must not rely on supervised extraction strategies. It is necessary to reduce the amount of training set to a bare minimum.
- Corpus Heterogeneity, because the corpus structure can differ a lot from corpus to corpus. Strategies like parsers and NER are not that useful because they are typically tailored to a specific corpus and can produce incorrect results when used in other corpus. More shallow parsing methods should be used, like, for example, Part Of Speech Taggers, to combat this problem.

- Scalability and efficiency, to be able to process large amounts of data in a reasonable time, the algorithm should be fast, and precise. [19] [20]. Today IE and Open IE are deeply connected, so behind this point, every time we speak about IE, we are also talking about Open IE.

2.2.1 IE applications

Clinical information extraction tools – Health providers can generate a lot of information about patients, diseases, treatment. With so much helpful information and with the help of IE, this information can be easily accessed by a health provider to help the patients better. This information can also be used to predict future complications and detect problems still undetected on the patients [21].

Information Extraction from Emails – Email is still an essential tool today, and in the corporate environment is still one of the most used methods of communication [22]. Unfortunately, people are also becoming overloaded by the amount of email received, so with the help of IE, a good amount of these emails can be filtered out as spam. IE can also be helpful to separate emails by topics (for example, bill, personal, promotions ...) [23]

Big Data information extraction– The improvement of technology (faster computers, better Internet access and others) creates a good amount of data. By IBM estimates, more than 2,500,000 Terabytes is created each day. This information can be structured, semi-structured or unstructured. Unfortunately, a good amount of this information is from the unstructured kind and is predicted to get even more in the future. Whit so much information that a system cannot understand, IE plays a significant role because it can transform that information into useful information that can be consumed and stored by a computational system [24].

2.2.2 IE history

We can say that IE had a peak in interest in 1980 when Defense Advanced Research Projects Agency (DARPA) started to fund competitive research in the IE field. However, even before that, there were other projects like the Linguistic String Project (1960-2005) [25] that was one of the early projects focused on computer processing of natural language and ELIZA (early 1970) that tried to create a connection between NLP and computer language [26].

In 1962 the association for Machine Translation and Computational Linguistics or AMTCL was created, and in 1968 changed its name to Association for Computational Linguistics(ACL). Each year ACL host a conference that produces and divulges paper regarding natural language and computation [27]. Acl is "the international scientific and professional society for people working on problems involving natural language and computation".

From 1987 to 1997, DARPA created and financed 7 Message Understanding Conference (MUC-1 to MUC-7) responsible for creating better information extraction methods.

Until 2007 most of the IE approaches used handwritten rules to extract information, but in 2007, a new paradigm was introduced, the open IE [19]. Unlike standard IE, open IE is not limited to pre-emptive knowledge about the data that will be analyzed.

Beginning in 2009 [28] the use of deep learning started to be used to try to automize the creation of datasets [29].

In 2018 a new paradigm appeared, the transformers approach that came to solve one of the problems of the previous techniques that is the unidirectionality and the limited choices of architectures that can be used during training [8]. The first significant system created was the BERT (Bidirectional Encoder Representations from Transformers) [13]

For example, transformers were used to extract important information in business documents (bidding and sales documents) using small training sets but getting high accuracy results deemed acceptable by real people. [30]

Also, in a study published in 2020, transformers were used to extract clinical concepts achieving state-of-the-art results that even surpassed results obtained using LSTM-CRFs models [31].

2.2.3 IE Pipelines

The figure 2.1 shows a generic pipeline. Depending on the problem, some blocks may be different or even excluded. For example, the block Sentiments relation extraction could be not necessary for some solutions. We decided to show this pipeline because it is extensive and similar to the one used in this dissertation.

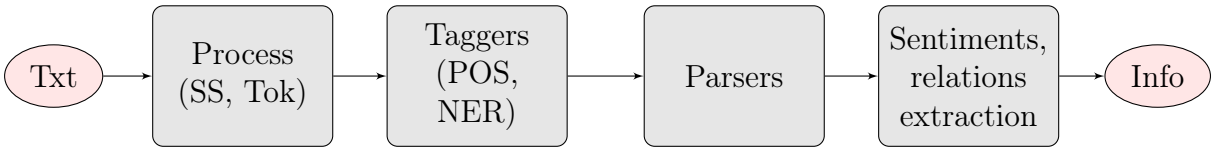


Figure 2.1: Information Extraction (IE) pipeline developed, showing the main processing blocks.

Process – Usually, the raw data is filled with several sentences that are composed of several words. Before any relevant data can be extracted, the input must pass by two processing methods, the sentence splitter and the tokenization:

- The sentence splitter, as the name says, divides the input into several sentences. Usually, this process uses punctuation to know if the sentence ended or not. This process usually is not an easy task, but nowadays, most sentence splitters can achieve state-of-the-art results.
- The tokenizer is used to divide each sentence into several tokens. Tokens usually are words, sub-words, numbers, punctuation or characters.

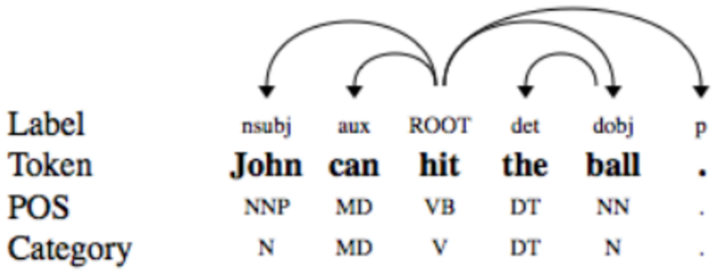


Figure 2.2: Example of how much information can be extracted after processing, tagging and parsing a sentence

---

Taggers – In the taggers category, we got both parts of speech and named entity recognition.

- POS module is responsible for tagging each token with is part of speech (noun, pronoun, verb, adjective, adverb, preposition, conjunction, and interjection are the English part of speech).
- Ner module is responsible for locating and classifying named entities. This process will be well explained later in this thesis.

Parsers– The parser module is responsible for determining the syntactic structure of the sentences. Basically, this is understanding how each word correlates with the other in a sentence. This process can only be made after doing the POS of each token. For example, in the figure 2.2 we can see the result of the pipeline so far.

Sentiments relation extraction– The Sentiments relation extraction module is responsible for understanding the author's general feeling related to a specific topic (a person, a place, an object or other subjects). To be able to do this step, the sentences should have been parsed. This method is going to be better explained later in the development part of the dissertation.

2.2.4 Representative IE tools

Waikato Environment for Knowledge Analysis (WEKA) [32] – The WEKA [33] software contains a collection of tools for data mining, data analysis and other areas of machine learning. The WEKA software also includes tools for processing data using IE.

General Architecture for Text Engineering (GATE) – GATE [34] is a set of tools developed by the University of Sheffield, and it is used for natural language processing tasks as information extraction in several languages. GATE also includes ANNIE (A

Nearly-New Information Extraction System), an agglomerate of modules such as tokeniser, Sentence splitter, semantic tagger, name matcher, and Gazetteer lockups.

Apache OpenNLP – Apache Opennlp is a toolkit library for processing natural language text. This library supports tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, language detection and coreference resolution.(https://opennlp.apache.org/)

WOE – WOE [35] or Wikipedia-based Open Extractor was one of the first systems that utilized Wikipedia to train an open information extractor.

TextRunner – TextRunner [19] was one of the first systems to extract information without the need for human output (this was one of the first OpenIE approaches).

OLLIE – OLLIE [36] or Open Language Learning for Information Extraction is an OpenIE approach that can be used to extract information when the target relations aren't known in advance.

2.3 Named entity recognition

Due to the relevance of this task for the work performed, this section will be dedicated to NER. In this section, we will show its significance, its history and its development.

NER/NERC (named entity recognition or named entity recognition classification or entity extraction or entity identification) is a vital task used in NLP pipelines to identify and extract entities in natural language text. These entities can then be aggregated in different categories: organization names, locations, times, types of food, etc. Depending on the type of information needed to be extracted, for example, if the information required is about restaurants, some important categories might be organization names, types of foods, places, times.

2.3.1 NER applications

Machine translation – Translation is the procedure of converting a dataset from a language to another. This process can be executed by a human or a machine (automatic machine translation). Named entity recognition is a powerful tool in this field because it can improve the reliability of the machine translation algorithm [37], but this can as well be a problem because if the NER is not well implemented can lead to mistakes in the translation and completely change the meaning of the dataset. [37] .

Information extraction – NER is a crucial step in the Information Extraction pipeline. Without it, a lot of essential information could not be extracted [17]. In some fields like medicine and others, it is even critical to have a NER in the process of Information Extraction [38].

Question and answering – Question and answering tools such as chatbots had been developed to answer questions asked by human beings [39]. NER is a crucial step to understand the answer and to find a possible response [40].

Automatic text Summarization – Automatic text Summarization is a tool used to briefly summarize all of the important information contained in raw data. This can be used when there is a lot of raw data information but little structured information [41] (like, for example, in the medical field).

Sentiment analysis – Sentiment analysis is a tool designed to understand people feelings about entities (such as companies, products, movies, music, among others). Today there is a vast amount of raw information on the web (tweets, comments, reviews), and a lot of useful information can be extracted using sentiment analysis [42].

Semantic search – NER is also an essential tool for semantic search [43] (creating services that understand their user's intentions better and give more personalized results).

2.3.2 NER history

NER is a critical step in extracting information from text. This technique was firstly used by Lisa F. Rau in 1991 [44] to extract company names from text. This paper was one of the first papers to show the importance of NER and how it could be implemented.

Between 1991 and 1995, this technique was also used in another eight English papers [45]. In 1996 the popularity of the technique increased after being used in "Message Understanding Conference 96" [44] to identify and mark the names of people, organizations and geographic locations in natural language text. In 1999 an article was written showing a technique that used a ruled based NER (Gazetteers) in combination with machine learning [46].

Over the years, the CoNLL (Conference on Computational Natural Language Learning) has been very important to the development and recognition of NER. It also shined a light on how the techniques have been developing over the years. For example, in the 2003 CoNLL conference, most of the participants chose to follow a machine learning approach, and the most popular was the Maximum Entropy Model [47]. Other countries also created their conferences. For example, Portugal created the HAREM contest (evaluation contest for named entity recognition in Portuguese started in 2004) that was the originator of the most important set of corpus for the Portuguese language [48].

In 2011 a new approach known as deep learning was used to create a NER. This new approach tried to create a way to understand natural language text using AI [49]. With the appearance of deep learning new techniques were developed [50] such as Deep Transfer Learning [51], Deep Active Learning [52], Depp Adversarial Learning [53], Deep Reinforcement Learning [54], Neural Attention and others.

Around 2018 a new branch of deep learning appeared, and it was the transformers. Today is still used and has been achieving state-of-art-results [55] [56].

2.3.3 Main NER Aproaches, Methods and Techniques

Depending on the problem that you wish to solve Ner can be implemented through several ways, that can be classified into:

Rule-based NER – We can label this as the human approach in which a set of handcrafted rules is made to identify and classify named entities. Depending on the problem, this method had a good chance to succeed. One of the main problems was that it was expensive and time-consuming to make. It was also difficult to port to other problems. Even so, depending on the problem it can be a good approach [57] [58].

Gazetteers – The gazetteers approach uses lists containing names of entities (for example, cities, places, organizations, peoples, etc.) to recognize and classify entities in the raw text provided. Another type of gazetteers can be the trigger gazetteers in which a keyword can be used to identify a possible entity (for example, Ms. can be used to identify that the following statement is a person) [59]. This approach is straightforward and can get pretty good results. Unfortunately, the creation and maintenance of the lists can be a challenging and tedious process [59] it also has problems with ambiguity (For example gate can be an object or a name of a person depending on the context).

Patterns or shallow parsing– The patterns approach uses the part of speech (e. g. verb or adjective) to gather and aggregate terms (for example, street, avenue, road are all aggregate together). After creating this list, information can now be extracted.

Machine learning – We can divide Machine learning into four main branches: supervised learning, unsupervised learning, reinforcement learning, and deep learning. The supervised learning method uses a training set (or a corpus) that was already manually labelled by experts(generally by a linguist). Then using this corpus, we can train text that is still not labelled. The unsupervised learning method consumes an untrained data set and extracts patterns from it. Contrary to the supervised method, this one doesn't need a training set that was already labelled. The reinforcement learning method uses agents to learn polices [60] that can be used to label an untrained data set. These agents are trained using a reward system. The deep learning method is inspired by the structure and function of the brain. A set of non-linear but straightforward methods are brought together (communicating with each other) to resolve problems that the simple techniques couldn't do in their own [61].

Hidden Markov Model (HMM) – Hidden Markov Model is a modelling statistical machine learning technique that was used to develop new NER algorithms extending the concept of Markov chains [62].

Support Vector Machine (SVM) – The SVM utilizes a binary data training set to classify an untrained data set [63]. This approach is based on a statistical

learning framework or theory. SVM is considered a very robust approach [63] and can achieve high accuracy even with a large untrained data set [64].

Maximum Entropy Markov Model (MEMM) – The MEMM is a "powerful tool for representing sequential data" and it has been used to create new Ner algorithms. [65]

Conditional Random Field (CRF) – CRF is a sequence modelling algorithm similar to NEMM but also takes into consideration future observations. This approach takes the best form NEMM (features are not independent form each other) and HMM (future observations). [66].

Neural nets – Neural network is a type of deep learning technique [67]

Transfer learning – Transfer learning is a deep learning algorithm used to gather knowledge obtained in a dataset to complete a task in a different dataset [68]. This algorithm was grown in popularity recently and has been heavily used, for example, in medicine [69], language [70] [71] and others.

2.3.4 Representative NER tools

CTexTools 2 – CTexTools [72] is a corpus published by North-West University that is primarily used to support the official South African languages (Afrikaans, English, South Ndebele, Xhosa, Zulu, Sesotho, Pedi, Setswana, Swazi, Venda, Tsonga).

FreeLing– Freeling [73] is an open-source language analysis. Besides NER, it also implements tokenization, MSD-tagging, syntactic parsing and lemmatization. The language supported are Catalan, English, Galician, Italian, Portuguese, Welsh.

OpenNLP – OpenNLP library is a machine learning tool used to process natural language text. Besides NER, it also implements Language Detector, Sentence Detector, Tokenizer, Document Categorizer, Part-of-Speech Tagger, Lemmatizer, Chunker, Parser.

Chapter 3

An Information Extraction System for eWOM in Portuguese

For this work, the engineering research method was adopted (proposal of a solution for a problem) . This chapter presents information regarding the proposed solution and how it was developed, starting with Requirements Engineering based on Personas and Scenarios. Then a system overview is introduced to show what each module can do and how it was implemented to create the solution used in the dissertation.

3.1 Problem Analysis and Requirements

3.1.1 Personas and Scenario(s)

3.1.1.1 Personas

We considered two main users:

- The first is the hotel manager that wants to see what his clients say about his hotel and see what he can upgrade to improve in his hotel.
- The second one is the user who likes to travel and chooses the best hotel that fits his needs.

Hotel manager Mr. Antonio Silva

Age: 45
Job: Hotel manager
Family: Married 2 childrens
Location: Aveiro, Portugal
Education: Master's degree in "Tourism Management and Planning"

Bio

Mr Antonio Silva is an expert that holds more than two decades of experience in the hotel industry. He joined the Hotel Lago in Aveiro in 2010 and since then has been the hotel manager. He is responsible for managing the day-to-day hotel operations and, to do so, has more than 40 capable hotel employees that fall under his responsibility. Mr Silva is also responsible for a new renovation program that will modernize all the rooms and create new spaces to entertain the clients.

Before this, he also worked in the Pasta hotel in Porto, where he gained a lot of experience working under a renowned hotel manager. Mr Silva holds a master's degree in "Tourism Management and Planning" from the University of Aveiro.

Goals

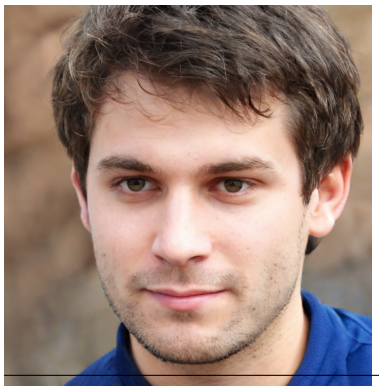
- Improving the hotel so the client can have a better time.
- Spend less time reading the comments about the hotel.

Frustrations

- Needing to read a thousand comments to see what he should improve in the hotel.
- Having to visit a dozen of sites to get all the comments about the hotel.
- Finding comments in languages he is not so familiar with.
- Not very tech-savvy, so some sites are not that easy to navigate to him.

Student Pedro Manuel

Age: 25
Job: Student
Family: Single
Location: Aveiro, Portugal
Education: Master's student



Bio

Pedro is a student at the University of Aveiro, where he is taking a master's degree in computer engineering. He lives in Aveiro, and he likes to travel, and most of his free time is used to planning for his next trip.

Goals

- Spending less time finding the cheapest hotel.
- Spending less time reading the comments to the advantages and disadvantages of a given hotel
- Finding an easier way of finding a comfortable bed in a given city.

Frustrations

- Spending too much time reading the comments trying to find information about the bed in the hotels.
- Information is too spread out on the internet.
- Some websites are biased about the information they spread.

3.1.2 Scenario 1

Mr Antonio Silva wakes up every weekday at 7 o'clock to get ready to manage his hotel. After waking up, he takes a shower and gets dressed as usual. While having his breakfast, he enters his hotel webpage at InfoXtration.com [**P0 – Access via webpage**] to check what people are saying about his hotel [**P0 – Process eWom about hotel**].

After logging in [**P1 – User accounts**], he finds that he got 20 new comments[**P2 – info on new comments**] since his last login. Of these 20 comments, he can see that five are bad and ten are good[**P0 – global sentiment analysis on individual comments**].

First, he decides to read the good ones, the first ones are speaking about the services and rooms (he is already used to getting good reviews on those), but the 8th is about the new fountain built last month.

Because it is the first time clients are speaking about that, he decides to view the full comment [**P1/2 – see selected full comment**] so he can read it in full and discover when and where it was made.

After that, he gets back to the front page to now read about the bad ones. He sees nothing new there because all the bad comments are about problems already known, and a solution is already being prepared, so the number of bad comments should decrease. Of all the bad ones, the most critical is the lack of a ramp on the secondary entrance to the hotel [**P0 – extract details**] (more than fifty per cent of the bad comments since last month) [**P0 – get percentages**] [**P1 – evolution over time**]. Still, it is already being built so he is confident that it should get better comments and if he is lucky a small raise (he hopes).

After checking on his hotel, Mr Antonio decides to check the ranking of his hotel in Aveiro. After changing the page, he can see that his hotel is in the 11th position with 246 good reviews and 5th place [**P1 – ranking based in automatic sentiment analysis**] with an 84 % ratio of good to bad comments.

3.1.3 Scenario 2

After finally finishing his exams, Pedro is now ready to start preparing for his summer vacation. He already chose that he wants to visit Porto, and after getting an awful experience last summer because of the hotel, he is now compelled to do better research, so he does not repeat the same mistake twice. After doing some research, he discovers a new website, compareHotel, that allows him to get all the hotels in a region[**P0 – Search by region**] and then compare them with each other.

The first thing he did when he entered the site was locking his region to Porto [**P0 – Search by region**], after doing that, he discovered that are more than 70 hotels[**P2 – Show region information**] on the Porto area. Then he ordered them by ranking. First, he tried by positive reviews[**P1 – Order by positive reviews**] and then by positive/negative reviews ratio[**P1 – Order by ratio**]. After doing this, he chose four hotels that were in the top 10 in both rankings.

After consulting the hotels site[**P0 – Show hotel information**] he discovers that one of them he cannot afford and another is already full, so the only ones left are hotel Turquia e hotel Flores. So once again, he decides to use the site to compare them[**P2 – comparison between specific hotels**]. Both hotels have a few negative reviews and a good amount of positive ones. So he decides to see what people dislike more about the hotels[**P0 – global sentiment analysis on individual comments**].

Most of the complaints in Flores hotel are about the noise [**P0 – list of negative comments order by most common**], it seems that the hotel is located near a famous night club so it can be a little noisy at night, to Pedro this is not a problem because it doesn't expect to sleep that much at night. Then he checks the Turquia hotel and discovers that most of the complaints are about the hotel's hidden fees.

After some consideration, Pedro decides to book in the Flores hotels. He decided not to choose the Turquia hotel because of the bad complaints about the hidden fees and the cleansing of the room.

3.1.4 Requirements

The requirements are an essential task in any project because it defines how your project will work, who will use it, and how they will use it. In this project, the requirements were separated between Functional and Non-Functional.

The functional requirement defines how a solution should behave and what information will it manage.

The non-functional requirements are not essential for the solution to work but are still crucial to the system. For example, in the table 3.2, we have the requirement "The system should periodically update his database with the new commentaries" the solution can function with only the data gathered during the first iteration of the crawler is used. Still, if the solution doesn't get more information periodically, the project will fail.

After analysing each scenario, a group of requirements was compiled. This step is very important because it defines what our program should implement and its results. The first table 3.1 represents the requirements extracted from both scenarios and the second table 3.2 represents the requirements derived from the first table.

3.1.4.1 From Scenario

After analysing each scenario, a group of requirements was compiled. The first table 3.1 represents the requirements that were extracted from both scenarios.

3.1.4.2 Derived

After gathering all the requirements derived from the scenarios, new requirements were created to fully support them. These new requirements can be found in table 3.2.

Table 3.1: Requirements from Scenario

	Requirement	P	Type
I1	Access via web	0	Interac.
F1	Process eWom texts	0	Func.
F2	User accounts	0	Func.
F3	Info on new comments	2	Func.
F4	Global sentiment analysis on individual comments	0	Func.
F5	See selected full comment	1	Func.
F6	Extract details	0	Func.
F7	Evolution over time	1	Func.
F8	Search by region	0	Func.
F9	Provide region information	2	Func.
F10	Order by positive reviews	1	Func.
F11	Order by ratio	1	Func.
F12	Provide hotel information	0	Func.
F13	Comparison between specific hotels	2	Func.
F14	List of negative comments order by most common	0	Func.

Table 3.2: Requirements derived from initial requirements

	Requirement	Derived from	P	Type
NF1	Database must be protected against sql injection	I1	0	NFunc.
NF2	The website must be online all the times	I1	1	NFunc.
NF3	The system should periodically update his database with the new commentaries	F3	0	NFunc.
NF4	The system needs to be fully operational while updating its database with new commentaries	NF3	0	NFunc.
NF5	The website should always show some input, even when it is fetching data	F1 to F14	0	NFunc.
F15	The system must have a fully functional NLP pipeline to process commentaries	F4	0	Func.
NF6	The website must have a user-friendly interface	F1 to F14	2	NFunc.

3.2 System Overview

The overall process is shown in Fig. 3.1. It is essentially a classic pipeline, starting with data gathering to get raw data, then doing sentence splitting and tokenisation. After these steps, part of speech and named entities recognition can be done. Then the dependency parser can be executed using all the information compiled before. All that is left is to do a sentiment analyse and store all the results in a database.

Data Gathering – This module is used the gather information from the hotel site (booking).

Data gathering is a crucial step in information extraction because without a good

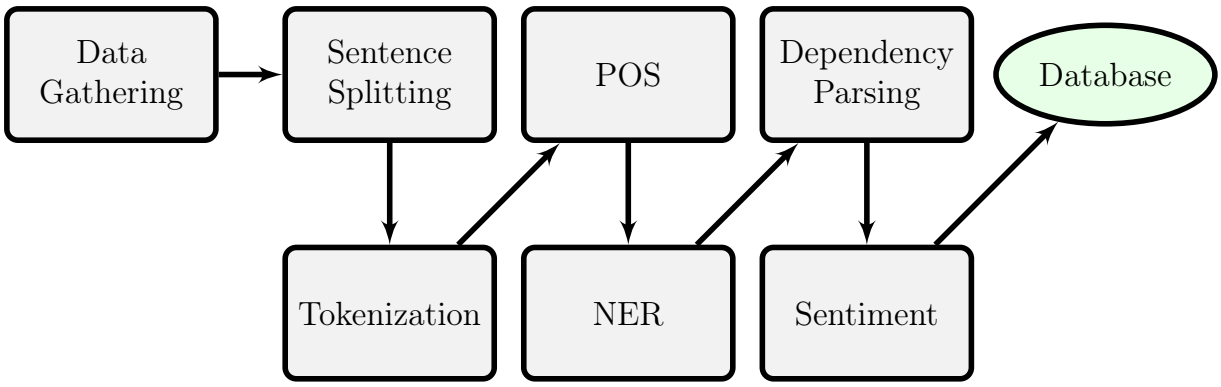


Figure 3.1: NLP generic pipeline

sample of raw data is difficult to extract enough relevant information. With the appearance of the internet, this information can now be found in great abundance, for example:

- Internet repositories (like books, papers, news and others).
- Social platforms (like Twitter, Facebook, blogs and others).
- Websites revolving around commentaries (such as IMDB, booking and others).
- Audio transcriptions.

This information can be gathered using API like Wikipedia or Babel. It can be downloaded, and web crawlers can also be used to strip a website of any raw data.

Sentence Splitting – This module is used to split raw data into sentences (this is done because the input of the dependency parser must be sentenced). The input of this module is the raw data gathered before, and the output is a list of all the sentences. For example, if we use the following raw data, we would get the following output.

Raw data:

```
"Infelizmente uma funcionária deu escândalo na frente de clientes
fazendo checkin só porque minha noiva pegou o abridor de garrafas
no balcão para abrir duas garrafas que compramos no posto proximo
ao hotel. Se estavamos errados em ter feito isso, faltou
profissionalismo da parte dela de ser mais cordial com o tratamento
conosco."
```

Output:

```
Infelizmente uma funcionária deu escândalo na frente de clientes
fazendo checkin só porque minha noiva pegou o abridor de garrafas
no balcão para abrir duas garrafas que compramos no posto proximo ao
hotel.

Se estavamos errados em ter feito isso, faltou profissionalismo da
parte dela de ser mais cordial com o tratamento conosco.
```

Tokenization – This module is used to split the sentences into tokens. This is a necessary step before doing part of speech. The input of this module is a sentence, and the output is a list of all the tokens contained in the sentence. For example, using the following input "Hotel bastante limpo e cuidado." we get the following output shown in the example 3.1.

■ **Example 3.1 :** The output of the tokenization module

```
"words": [{
            "indice": 1,
            "form": "Hotel",
        }, {
            "indice": 2,
            "form": "bastante",
        }, {
            "indice": 3,
            "form": "limpo",
        }, {
            "indice": 4,
            "form": "e",
        }, {
            "indice": 5,
            "form": "cuidado",
        }, {
            "indice": 6,
            "form": ".",
        }
    ]
```

Part of Speech Tagging – This module is used for grammaticalising the tokens (noun, verb, adjective) and the lemmatization (grouping together the inflected forms of a word) of the tokens. The lemmatization is a necessary step so that later sentimental analysis can be done.

The input is the list of the tokens, and the output is the lemma and the POS tag (the difference between "tag" and "CPostag" is that tag is a more specific POS tag and the "CPostag" is a more widespread tag see an example of output).

■ **Example 3.2 :** Output using the information extracted in the tokenizer

```
"words": [{
            "indice": 1,
            "form": "Hotel",
            "lemma": "hotel",
            "tag": "n",
            "CPostag": "n"
        }, {
            "indice": 2,
            "form": "bastante",
            "lemma": "bastante",
            "tag": "adv",
            "CPostag": "adv"
        }, {
            "indice": 3,
            "form": "limpo",
            "lemma": "limpo",
            "tag": "adj",
            "CPostag": "adj"
        }, {
            "indice": 4,
            "form": "e",
            "lemma": "e",
            "tag": "conj-s",
            "CPostag": "conj"
                }, {
            "indice": 5,
            "form": "cuidado",
            "lemma": "cuidado",
            "tag": "n",
            "CPostag": "n"
                }, {
            "indice": 6,
            "form": ".",
            "lemma": ".",
            "tag": "punc",
            "CPostag": "punc"
        }
```

```
38      ]
```

Dependency parsing – The input for this module is the output of the POS 3.2 module, and the output is the token (with all of the previous information), the dependencies (if the dependency is 0, it means that the token is the root of the sentence) and a list of the dependents.

This module is used to create a structure in the sentence. With this information, relations between the tokens can be made. For example, in the sentence "The room is big." the verb "be" is the root of the sentence, and the token "big" is dependent on the token "room", with this information, we can characterize the room as big. The next image 3.2 is a representative example of how the phrase "John can hit the ball." is processed.

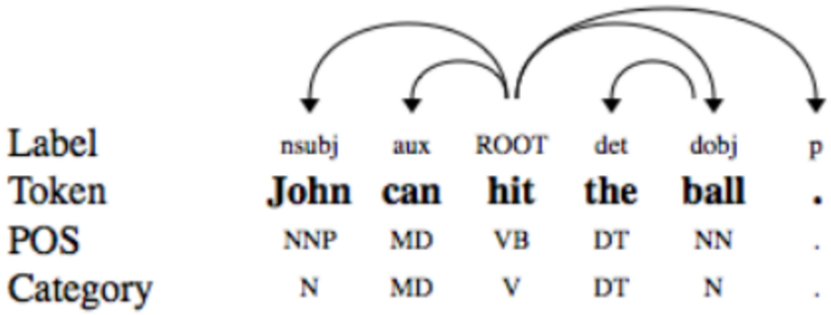


Figure 3.2: Example of using dependency parser in the phrase "John can hit the ball".

Named Entity Recognition – This module is used to find named entities ("Mono or multi-word expression belonging to a potentially interesting class for an application" [74]) that are relevant for the theme that is being explored. In the following example 3.3, we can see three possible lists containing named entities.

■ **Example 3.3 :** Examples of a list of Named entities

```
1 Locations{Porto, Aveiro, Lisboa};
2 Names {Marco, David, Paulo };
3 Organization {Facebook, Google, Amazon }
```

Sentiment Analysis – The objective of this module is to try to understand the feelings/opinions people have regarding a topic (for example, this can be a person, a place or a company).

To characterize each named entity, we need to use the dependencies created before and a list containing information about the feeling each word express (for example, "nice" as an adjective is a good opinion, but "awful" as an adjective is a bad opinion).

In the example 3.6, we can see that both "limpo" and "cuidado" came as positive and because "hotel" is connected to "limpo", we now know that "hotel" "limpo" is a positive sentence about a location (NER tag).

■ **Example 3.4 :** Input of the sentiment analyse module

```
Hotel bastante limpo e cuidado.
O quarto é confortável e prático e a zona exterior é simpática.
Achei a limpeza horrivel.
```

■ **Example 3.5 :** Output of the sentiment analyse module

```
Hotel bastante limpo(bom) e cuidado(bom).
O quarto é confortável(bom) e prático(bom) e a zona exterior é ...
   simpática(bom).
Achei a limpeza horrivel(mau).
```

Semantic Knowledge base – This module objective is to store all the information in a way that is easier to access and query. The semantic knowledge base is the process of storing and processing knowledge. Contrary to a regular database (a collection of tables and cells filled with data), knowledge bases try to achieve an organized collection of data similar to how the human brain stores information. Knowledge bases information is stored as classes subclass relationships and instances (ontology). In the following figure 3.2, we can see an example of a semantic database structure.

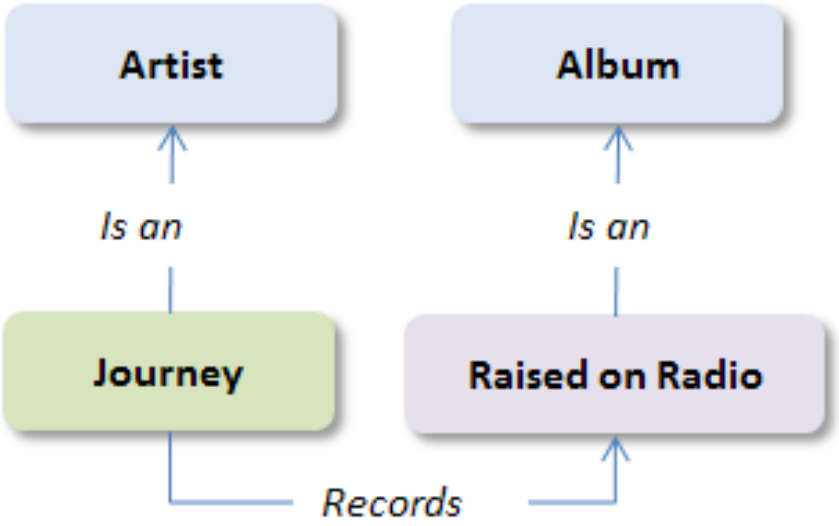


Figure 3.3: Example of semantic database stored data

3.3 Implementation

Data gathering – Booking was selected as a source because it had a large amount of data and had a simple structure.

Jsoup, which is a Java HTML Parser, was used to extract the information necessary from each hotel. In this module, we only used and tested Jsoup because it gave us all the essential information, and the only improvement that could be possible was timewise. Still, even on a domestic computer, it only took around 30 minutes to gather the information of 1,974 hotels.

Unfortunately, using an API to get information from Booking or other relevant sites was not possible because they either did not exist, were a paid service, were only for hotel owners or couldn't be used in research.

The gathering of information from Booking using Jsoup was divided in 4 steps: (fig 3.4)

Getting all the relevant locations used on the website. The district webpage was used and then from each district, all the locations names and URLs were gathered (580 different locations were gathered using this method).

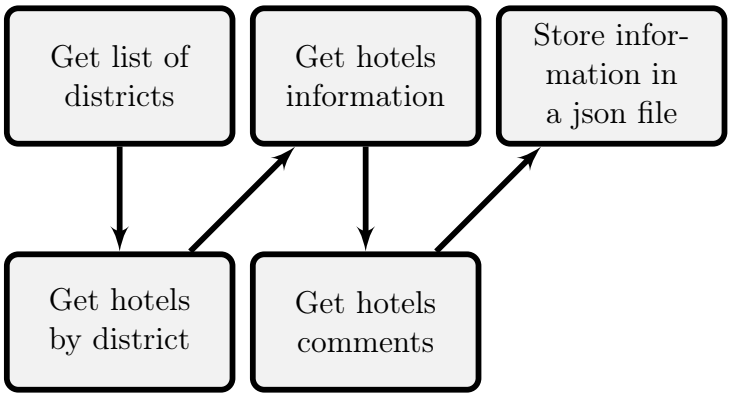


Figure 3.4: How the data was gathered in booking using a crawler

■ **Example 3.6 :** Example of three locations extracted from booking

```
espinho
http://www.booking.com/destinationfinder/cities/pt/espinho...
penela
http://www.booking.com/destinationfinder/cities/pt/penela...
furnas
http://www.booking.com/destinationfinder/cities/pt/furnas..
```

Getting all the hotels names (example 3.7) using the locations that was gathered in the last step.

■ **Example 3.7 :** Example of three hotels, their names and URL

```
casa-de-soudos
http://www.booking.com/hotel/pt/casa-de-soudos...
oporto-trendy-heritage
http://www.booking.com/hotel/pt/oporto-trendy-heritage...
grande-bom-jesus
http://www.booking.com/hotel/pt/grande-bom-jesus....
```

This was done using the crawler and the URLs collected to get all the relevant links on that page using patterns 3.8.

■ **Example 3.8 :** Example of three hotels, their names and URL

```
Pattern hoteisPatern = Pattern.compile("/hotel/pt/");
Pattern paginaSeguinte = Pattern.compile("Pagina seguinte");
```

Then the next step was getting the relevant information about the hotel, such as the

location and the hotel name (the difference between the name and the actual name is that the name is the identifier used in the URL and the exact name is the name Booking uses to identify the hotel).

■ **Example 3.9** : Important information that characterizes each hotel.

```
{
        "name": "park-italia-flat",
        "url": ...
           "http://www.booking.com/hotel/pt/park-italia...",
        "location": "Praceta Adelino Amaro da Costa 77...
           2, 4050-012 Porto, Portugal",
        "realName": "Park Italia Flat"
    }, {
        "name": "sunshine-apartment",
        "url": ...
           "http://www.booking.com/hotel/pt/sunshine....",
        "location": "Quinta do Romão Clube Apart ...
           Algarve- Apartment Sunshine 105, 8125-301 ...
           Quarteira, Portugal",
        "realName": "Sunshine Apartment"
    }
```

Then the URL was used to get all the commentaries from each hotel. To achieve this result, the element function of Jsoup was used, as can be seen in the example 3.10.

■ **Example 3.10** : How each comment was gathered

```
Document doc;
// If a field was not present the default value that we store in ...
   the json files was - that represented no information
String neg = "_";
String pos = "_";
String date = "_";
String country = "_";

try {
    doc = Jsoup.connect(dest).get();
    Elements elements = doc.body().select("*");
    //all the elements from a page were extracted and then only ...
       the important one were stored
    for (Element element : elements) {
                        //
        if (element.className().equalsIgnoreCase("review_neg")) {
            neg = element.ownText();
```

```
        } else if ...
            (element.className().equalsIgnoreCase("review_pos")) {
            pos = element.ownText();
        } else if (element.className().equalsIgnoreCase( ...
            "review_item_date")) {
            date = element.ownText();
        } else if (element.className().equalsIgnoreCase( ...
            "reviewer_country")) {
            country = element.ownText();
        }
        op.add(new Opinion(count, country, date, neg, pos));
    }
}
```

The following example 3.11 shows an example of the extracted commentaries.

■ **Example 3.11** : Example of the information gathered from a hotel including commentaries

```
{
    "name": "a-casa-do-ouvidor",
    "op": [{
            "id": 0,
            "natio": "_",
            "date": "15 de Março, 2016",
            "badComent": "_",
            "goodComent": "Localização"
        }, {
            "id": 1,
            "natio": "_",
            "date": "9 de Dezembro, 2015",
            "badComent": "Do mau tempo.",
            "goodComent": "De tudo, donos muito ...
              simpáticos e prestáveis. ...
              Ofereceram-nos o pequeno-almoço."
        }, {
            "id": 2,
            "natio": "_",
            "date": "4 de Outubro, 2015",
            "badComent": "_",
            "goodComent": "Fantástico!! Acordar ...
              naquele sítio é mágico!"
        }, {
```

```
            "id": 5,
            "natio": "_",
            "date": "26 de Setembro, 2014",
            "badComent": "Gostei de tudo. Não tenho ...
               críticas.",
            "goodComent": "A casa é bonita, nova, ...
               decorada com bom gosto, com uma vista ...
               linda do mar e da Ilha de São Jorge ...
               uma piscina e uma Jacuzzi e com um ...
               anfitrião muito prestativo."
        }
    ],
    "local": "Estrada Regional Sao Miguel Arcanjo, 994...
       0-335 São Roque do Pico, Portugal",
    "realName": "A Casa do Ouvidor"
}
```

Sentence Splitting – Due to its simplicity, being available and being possible to train systems with state-of-the-art performance. Apache OpenNLP library [75] was used to split comments into sentences. To train the system Bosque Portuguese corpus was used [76] (code 3.12).

■ **Example 3.12** : Training the sentence splitter using bosque

```
opennlp SentenceDetectorTrainer -model pt-sent.bin -lang pt -data ...
   ptBosque.train -encoding UTF-8
```

In the following code (code 3.13), the corpus that was already trained is loaded in an input stream and added to a sentence model. Then using that model and sentence to divide as input, a list of sentences is returned.

■ **Example 3.13** : Code used for implementing Sentence Splitter

```
// Portugues corpus based in Bosque
InputStream is = new FileInputStream("NlpFiles/pt-sent.bin");
String sentences[];
SentenceModel model = new SentenceModel(is);
SentenceDetectorME detector = new SentenceDetectorME(model);
//list of sentences
sentences = detector.sentDetect(sentence);
is.close();
return sentences;
```

Tokenization– Like in sentence splitting to tokenize each sentence, the Apache OpenNLP library was also used.

In the following code (code 3.13), the corpus that was already trained is loaded in an input stream and added to a token model. Then using that model and sentence to tokenize as input, a list of tokens is returned.

The Bosque Portuguese corpus was used [76] (code 3.14) to get a helpful corpus that the algorithm could use.

■ **Example 3.14 :** Training the Tokenizer using bosque

```
opennlp TokenizerTrainer -model pt-token.bin -alphaNumOpt -lang ...
    en -data ptBosque.train -encoding UTF-8
```

Using the trained corpus, it was now possible to implement the tokenizer (code 3.15).

■ **Example 3.15 :** Code used to implement the tokenizer

```
// Portugues corpus based in Bosque
InputStream is = new FileInputStream("NlpFiles/pt-token.bin");
TokenizerModel model = new TokenizerModel(is);
TokenizerME tokenizer = new TokenizerME(model);

//list of tokens
String tokens[] = tokenizer.tokenize(sentence);
is.close();
return tokens;
```

Part of Speech Tagging– After searching for POS taggers and initial experiments and tests with some POS (for example, the Stanford NLP), Treetager was adopted to do the POS tagging and the lemmatization of each token contained in a sentence. There was no necessity for training a corpus to use treetager because they already had a pre-trained corpus using the Bosque. A module for POS tagging was created using the pre-trained corpus and the output from the tokenization.

As we can see in the following code (code 3.16), first a model is created using the pre-trained corpus, then using a list of tokens as input, we get a list containing the tokens it positions the lemmatization and the POS.

■ **Example 3.16 :** Part of speech tagging simplified code

```
tokenList=new ArrayList<>();
TreeTaggerWrapper tt = new TreeTaggerWrapper<String>();
// Portugues corpus based in Bosque
tt.setModel("TreeTagger\\lib\\portuguese-v2.par:UTF8");
tt.setHandler(new TokenHandler<String>() {
```

```
    public void token(String token, String pos, String lemma) {
            tok = new Token(token, pos, lemma, map);
            tokenList.add(tok);
            }
});
//tokens is the list containing all the tokens
tt.process(asList(tokens));
tt.destroy();
return tokenList;
```

Dependency parsing– Due to previous positive experiences with Maltparser[77] and because it has achieved state-of-the-art accuracy [78], it was adopted to implement the dependency parser module. First, a corpus needed to be trained using the tutorial given by the Maltparser website and the Bosque corpus.

Then the dependencies were made using the generated corpus and the output of the POS tagging module.

The generated corpus needed to be altered slightly because TreeTager outputted tags had different connotations than the generated corpus.

■ **Example 3.17** : Dependency parser simplified code

```
MaltParserService service = new MaltParserService();
            // Inititalize the parser model 'model0' and sets the ...
               working directory to '.' and sets the logging file ...
               to 'parser.log'

            service.initializeParserModel("-c model0 -m parse -w ...
               . -lfi parser.log");

            String[] tokens = new String[tokenList.size()];

            //retrieves important information about the tokens ...
               that was processed before
            for (int i = 0; i < tokenList.size(); i++) {
                tokens[i] = ....
            }
            graph = service.parse(tokens);
            service.terminateParserModel();
```

Named Entity Recognition– The first tool used to try to implement a NER module was Rembrandt [79]. Rembrandt was a tool created and implemented by Nuno Cardoso in 2008. This tool used the DBpedia and Wikipedia to create lists of named entities. After implementing the code and running it on top of our hotel commentaries, we concluded that this tool was not the best to be used in our implementation. This

was because this tool highlighted numbers and words that start with a capitalized letter(person names, locations, institutions).

Numbers were not relevant for our solution, and capitalized letter wasn't either. When studying the solution output, we noticed that really few entities were recognized. Almost all of them were locations like "Porto", "Aveiro" this information was not that useful because the hotel location was already known . Also areas outside the hotel were not relevant to characterize the hotel (for example, knowing that someone visited a museum when staying at the hotel was not pertinent to describe the hotel).

After using Rembrandt, the following solution was using the Wikipedia API to create a list of NE but instead of only using words with the first letter, capitalizing every word that was tagged as a noun was used. Using the Wikipedia API created an overflow of information, most of it not crucial for or solution.

For the final solution, and the one that was used, all the nouns were gathered, and then a list with the most used ones was created. Then taking the most relevant ones, five lists were created:

Gazetteer	Elements
Hotel locations	*hotel, quarto, garagem, balcão, elevadores, recepção, restaurante, estacionamento, bar, wc* hotel, room , garage, counter desk, elevators, reception, restaurant, parking lot, bar, toilet
Hotel features	*barulho, conservação, conforto, simpatia, disponibilidade, comodidade, ruído, cheiro, vista, custo, preço, profissionalismo* noise, conservation, comfort, friendliness, availability, convenience, fuss, smell, view, cost, price, professionalism
Hotel objects	*cobertor, lençol, minibar, tomada, mobiliário, televisão, toalha, espelho, banheira , cama* blanket, sheet, minibar, plug, furniture, television, towel, mirror, bathtub, bed
Services	*buffet, jantar, almoço, pequeno-almoço, atendimento, manutenção, limpeza, reserva* buffet, dinner, lunch, breakfast, service, maintenance, cleaning, reservation
Staff	*empregado, funcionário, pessoal, Staff, recepcionista* employee, clerk, individual, Staff, receptionist

Table 3.3: List of Named entities

Sentiment Analysis– To do sentiment analyse of the hotel commentaries, the SentiLex-PT 02 [80] lexicon was used [1]. This lexicon was composed of 4,779 adjectives, 1,081

[1]When this solution was created, there wasn't a lot of resources useful to do sentiment analysis, and the best one was SentiLex-PT. Nowadays there are already other solution like [81]

nouns, 489 verbs, and 666 idiomatic expressions. This lexicon was used to categorize (when possible) every NER token, using the entities relation and the lexicon. For each NER entity, a list of dependencies was gathered, then for each of them, they were categorized by the lexicon. To do this was used the lemma of the word and the POS of the token was. For each token, there were four possible outcomes, the word didn't exist in the lexicon, the term was defined by a positive sentiment (annotation 1), the word was characterized by a negative sentiment (annotation -1), and the word was ambiguous (annotation 0, for example, the word "uniform").

■ **Example 3.18 :** Example of the usage of the sentiment analyze module

```
Original comment: "Quarto acolhedor e cama confortavel."
Ner token : quarto
List of dependencies: [2]
Sentiment analyze: acolhedor(1)

Original comment:  "Hotel bastante limpo e cuidado."
Ner token : hotel
List of dependencies: [3, 6]
Sentiment analyze: limpo(1) cuidado (1).
```

Semantic Knowledge base – Jena apache was used to create the information to be stored in the semantic database, [82]. It was responsible for creating triples and then storing them using an ontology created and maintained in protégé [83].

In the following image (fig 3.5), we can see how the ontology was represented. A district can have several hotels, and each hotel can have several comments mad about it. Then each comment can have several subjects (the words that the NER module considered worth tagging), and each subject can have one or more adjectives describing it. The hotel has a location associated with him. Each comment has a country (the country of the person who commented) and the date of when it was made. Each adjective was a qualifier attached to it (this can be good, bad or indifferent).

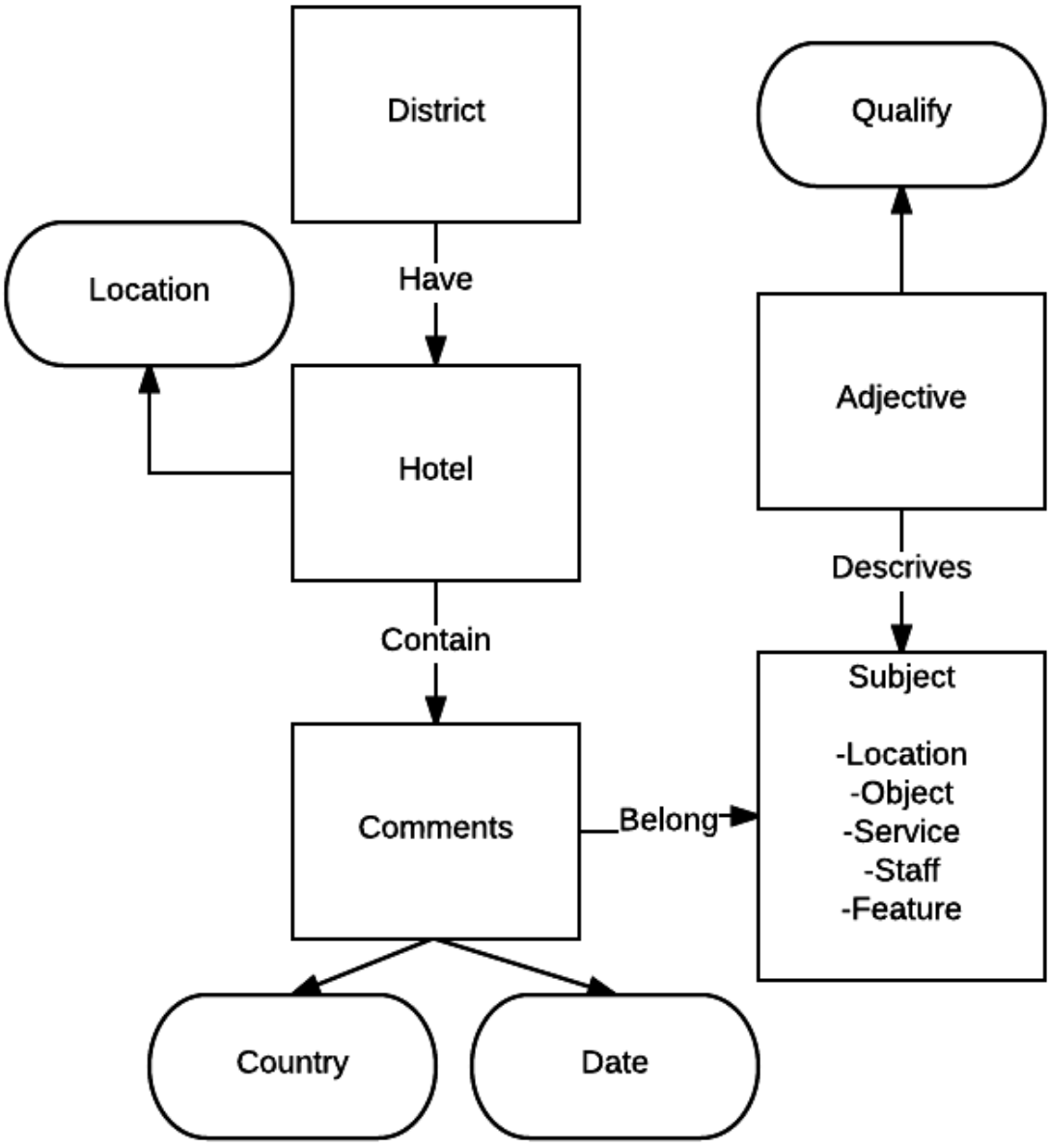


Figure 3.5: The ontology used in the dissertation

3.3.1 Software Structure

This chapter is meant to show and explain the structure of the solution. To do this, we will show the main classes and their primary functions.

3.3.1.1 Main packages

This first UML 3.6 shows the main packages of our solution.

The booking crawler is responsible for getting all the information required from booking and storing it in JSON files.

The NLP is responsible for extracting all the necessary information gathered by the crawler. This matter will be better explained in the next section.

The Semantic Database is responsible for storing all the triples in a semantic database and for doing all the required queries to get the relevant information out of the database.

The thread manager is an auxiliary package responsible for the management of the threads used in the crawler and on the NLP package.

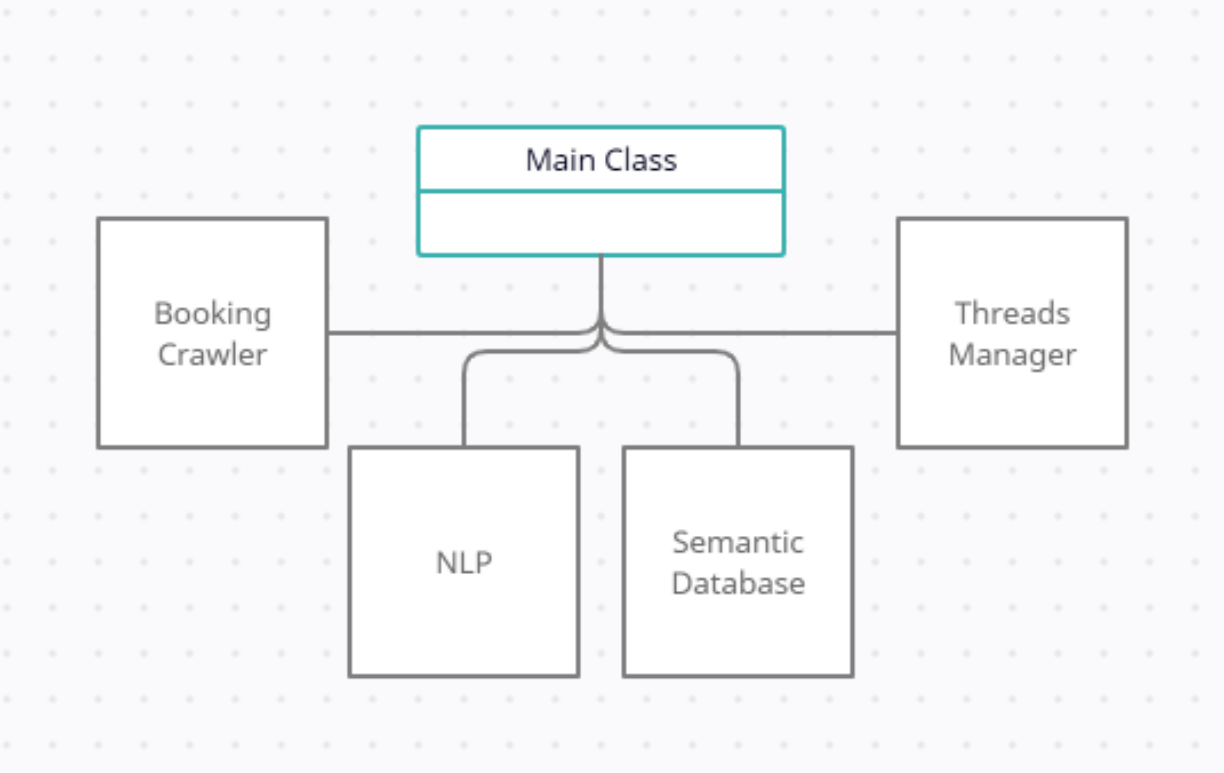


Figure 3.6: Structure of the solution

3.3.1.2 NLP package

In this UML, we can see that the package NLP 3.7 contains seven distinct java classes.

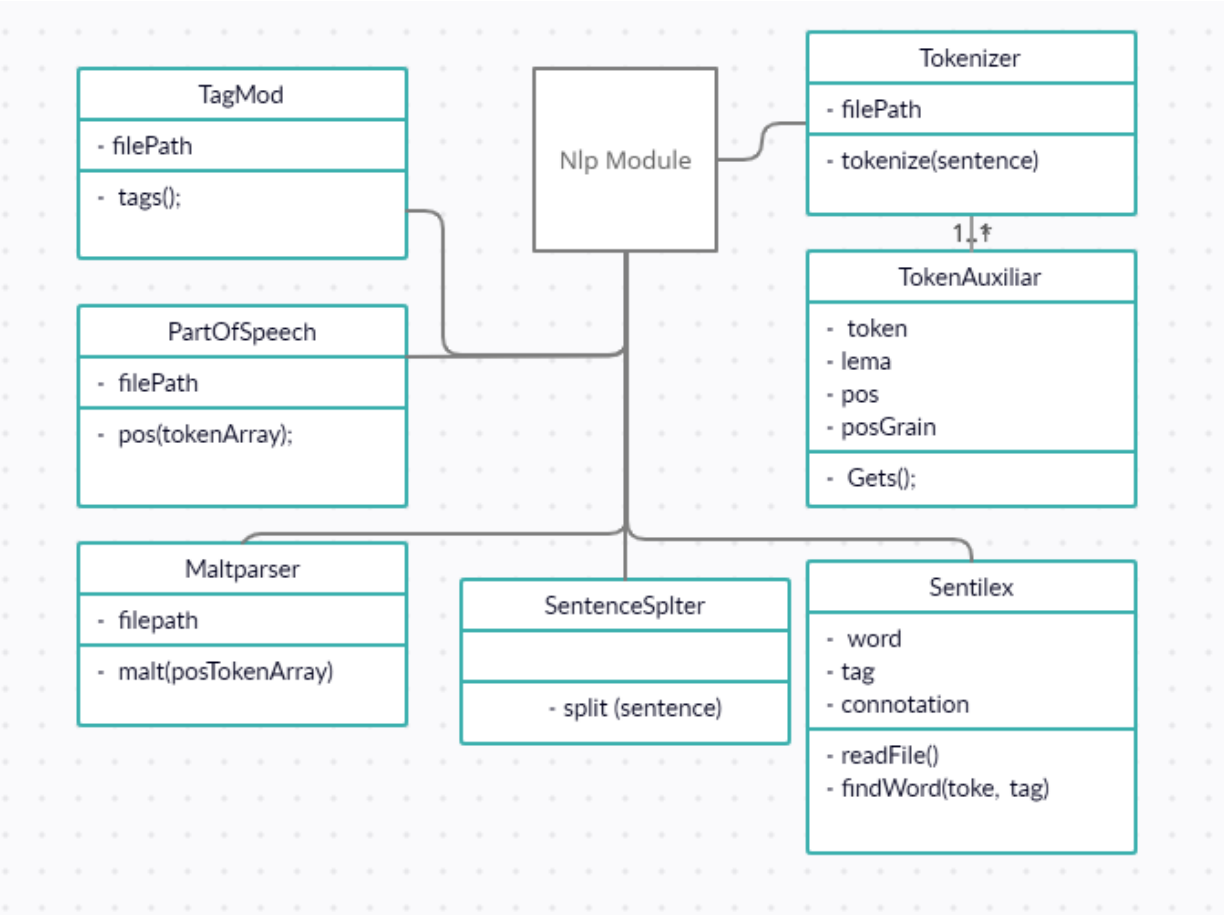


Figure 3.7: Java UML package describing the NLP module

The class SentenceSpliter receives a string containing all the sentences and returns a string array containing all the different sentences.

The class Tokenizer receives a sentence and returns an array containing all of the tokens derived by that sentence. The tokenizer has an auxiliary class that stores all the information relevant to defining a single token.

The class TagMod is used to modify the tags from TreeTagger to be used by the maltParser. It receives the output of the TreeTagger and creates the input for the Maltparser.

The class PartOfSpeech implements the TreeTagger. It receives an array containing all the tokens of a single sentence and returns a list of tokens that now contains the POS and the lemmatization.

The class Sentilex reads the sentimental analysis and searches for any particular token on the sentimental analysis list. If that token exists, it returns if it is a good, bad or neutral sentiment.

Chapter 4

Results

This chapter is responsible for showing some examples of the data gathered and of the information that was extracted from that data.

This chapter will also show some statistics about the data gathered and how the information extracted and the pipeline used was also beneficial to other projects outside this dissertation.

4.1 Examples

The first information that was extracted from booking was all the regions of Portugal that had hotels. The information record was the location name and the respective URL. In the example 4.1, we can see a snippet of that information.

■ **Example 4.1 :** Example of hotels and their regions

```
sao-bartolomeu-da-serra
http://www.booking.com/destinationfinder/cities/...
pt/sao-bartolome...

costa-de-caparica
http://www.booking.com/destinationfinder/cities/...
pt/costa-de-caparica....

ervidel
http://www.booking.com/destinationfinder/cities/...
pt/ervidel....

azurara
http://www.booking.com/destinationfinder/cities/...
pt/azurara...
```

```
16
17 vila-praia-de-ancora
18 http://www.booking.com/destinationfinder/cities/...
19 pt/vila-praia...
```

After getting all the regions, each region was processed, and all the hotels of that region were extracted. In the example 4.2, we can see a snippet of that information. .

Example 4.2 : Example of information gathered about the hotels

```
1 {
2         "name": "park-italia-flat",
3         "url": ...
             "http://www.booking.com/hotel/pt/park-italia...",
4         "location": "Praceta Adelino Amaro da Costa 772, 4...
             050-012 Porto, Portugal",
5         "realName": "Park Italia Flat"
6     }, {
7         "name": "sunshine-apartment",
8         "url": ...
             "http://www.booking.com/hotel/pt/sunshine....",
9         "location": "Quinta do Romão Clube Apart Algarve- ...
             Apartment Sunshine 105, 8125-301 Quarteira, ...
             Portugal",
10        "realName": "Sunshine Apartment"
11    }
```

After getting all hotel information, that information was used to extracted all the commentaries of each hotel. In the example 4.3, we can see a snippet of that information.

Example 4.3 : Example of information gathered about the hotels with comments

```
1   "name": "Naturmar Praia",
2     "op": [{
3             "id": 0,
4             "natio": "_",
5             "date": "19 de Julho, 2015",
6             "badComent": "Apartamento antiquado, ...
                 torneiras c pouca agua, ambiente ...
                 envolvente.",
7             "goodComent": "Da Localização."
8         }, {
```

```
            "id": 1,
            "natio": "_",
            "date": "1 de Janeiro, 2016",
            "badComent": "_",
            "goodComent": "Da simpatia do sr. Daniel, ...
              deixou-me antecipar o check-in e fazer o ...
              check-out quando quisesse. Obrigado sr. ...
              Daniel"
        }, {
            "id": 2,
            "natio": "_",
            "date": "20 de Agosto, 2015",
            "badComent": "deviam investir um pouco mais ...
              na decoração já está fora de moda e o ...
              ambiente também é importante",
            "goodComent": "facilidade de me deslocar sem ...
              carro visto ter tudo perto"
        },
        {...}

        ]
```

After getting all this information and using the pipeline described before, all the commentaries were sentence split, and each sentence was then separated by tokens. After going through this process, each comment was also analysed to extract the NER and then passed through a POS module and a dependency parser module. In the image 4.1 we can see the raw output from Maltparser when feeding the sentence ´´Adorei Este Lugar recomendo Uma visita.".

```
1 [I:0(0->1 DEPREL:STA )][O:3,4,6,7]ID:01 FORM:Adorei LEMMA:adorar CPOSTAG:v POSTAG:v-fin FEATS:_
2 [I:3(3->2 DEPREL:>N )][O:]ID:11 FORM:este LEMMA:este CPOSTAG:pron POSTAG:pron-det FEATS:_
3 [I:1(1->3 DEPREL:ACC )][O:2]ID:21 FORM:lugar LEMMA:lugar CPOSTAG:n POSTAG:n FEATS:_
4 [I:1(1->4 DEPREL:P )][O:]ID:31 FORM:recomendo LEMMA:recomendar CPOSTAG:v POSTAG:v-fin FEATS:_
5 [I:6(6->5 DEPREL:>N )][O:]ID:41 FORM:uma LEMMA:um CPOSTAG:art POSTAG:art FEATS:_
6 [I:1(1->6 DEPREL:ACC )][O:5]ID:51 FORM:visita LEMMA:visita CPOSTAG:n POSTAG:n FEATS:_
7 [I:1(1->7 DEPREL:PUNC )][O:]ID:61 FORM:. LEMMA:. CPOSTAG:punc POSTAG:punc FEATS:_
```

Figure 4.1: Raw output from MaltParser

In the following table 4.1 we can see the result with all the relevant data of this example"Funcionários atentos aos detalhes.".

In Fig.4.2, we can see the relationship between words.

As we can see, the word "funcionários" was tagged by the NER as an essential word, and then using the relationships between tokens is known that "funcionários" is connected

Table 4.1: Processing of a sentence

Word	Lemma	Entity	POS tag	CPostag	Dependencies	dependents
Funcionários	Funcionários	staff	n	n	-	[2,3,4]
atentos	atento	-	adj	adj	N\u003c	[ ]
aos	ao	-	prepxdet	prepxdet	N\u003c	[ ]
detalhes	detalhe	-	n	n	N\u003c	[ ]
.	.	-	punc	punc	-	

Figure 4.2: Maltparser relationships

to “atento” and using the Sentilex file, it can be deferred that “atento” as an adjective is considered a good word.

Then all this information is finally saved in the semantic database using triples. In the folowing image 4.3 the example "Hotel bastante limpo e cuidado." and the example “Funcionários atentos aos detalhes.” is recorded in the database.

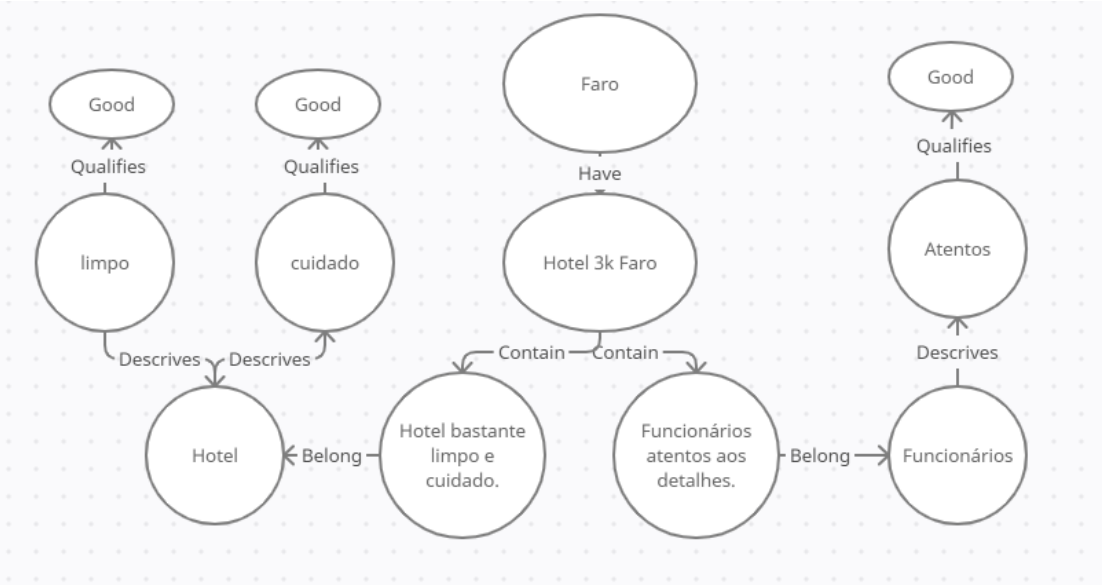


Figure 4.3: Semantic Database

4.2 Statistics

To populate our database with data, we extracted information about 11,647 hotels that belonged to 581 different regions.

From those hotels, two hotels stood out with 900 different commentaries in a spawn of three years. Those hotels were the "Moov Hotel Évora" and the "HF Fenix Lisboa".

In the figure 4.4, we can see the distribution of the commentaries made to the hotel "Moov Hotel Évora" over time.

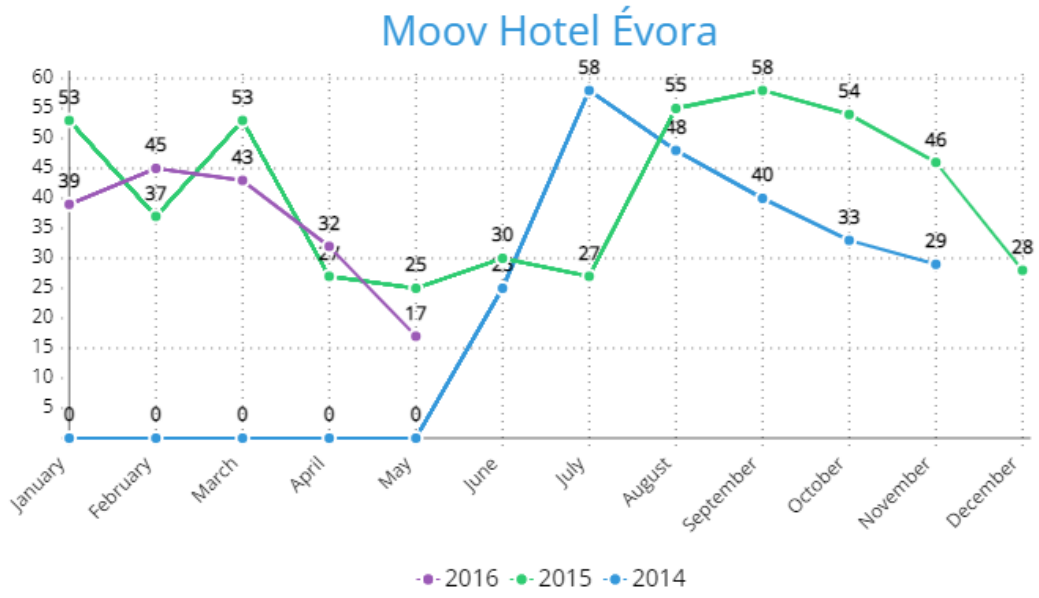


Figure 4.4: Distribution of commentaries made about Moov Hotel over time

---

Due to the big amount of data and the time it takes to process the next statistics will be of a sample of hotels. This sample contains 113 hotels with a combine amount of 12776 comments.

From those 113 hotels:

- There were 195 commentaries that criticized something about the hotel.
- There were 927 commentaries that said something good about the hotel.
- There were 175 commentaries that were neutral.

In the following figure 4.5, we can see the main entities that people criticize or speak good about.

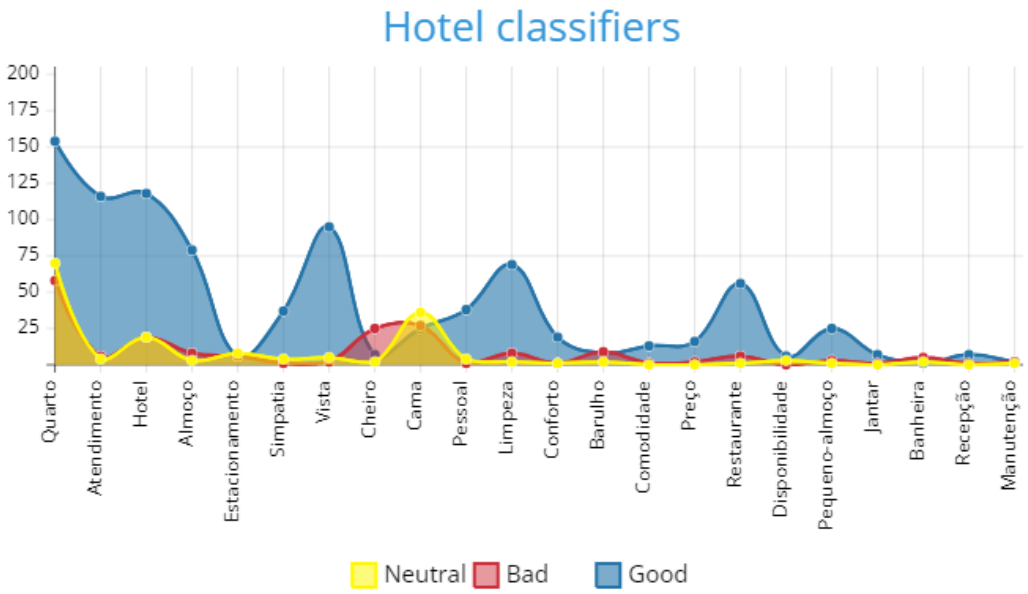


Figure 4.5: Graph showing the classification of entities

4.3 Use in a data2tex system

During the development of the data2tex, part of the information extracted by the system developed in the dissertation was used to help automatically create sentences regarding hotels [84]. The information that was passed was the hotel name, its location, the number of commentaries about that hotel, and the sentimental analysis results.

Example of the information that was sent to the data2tex project can be seen on the example 4.4 . In the example 4.5, we can see how the input influenced the output .

■ **Example 4.4 :** Snippet of the information send

```
{
      "name": "A Bela Piscosa",
      "local": "Trav. Xavier Da Silva, 2,4,6, 2970-794 ...
         Sesimbra, Portugal",
      "comentaryNumber": "65",
      "comentary": [{
         "name": "simpatia",
         "bad": [],
         "good": ["charmoso",
         "inexcedível"],
         "neutral": []
      },
```

```
        {
            "name": "vista",
            "bad": [],
            "good": ["maravilhoso"],
            "neutral": []
        },
        {
            "name": "cheiro",
            "bad": ["mau"],
            "good": [],
            "neutral": []
        },
        {
            "name": "cama",
            "bad": [],
            "good": [],
            "neutral": ["confortável"]
        },
        {
            "name": "pessoal",
            "bad": [],
            "good": ["simpático"],
            "neutral": []
        },
        {
            "name": "limpeza",
            "bad": [],
            "good": ["bom"],
            "neutral": []
        }]
    }
```

■ **Example 4.5 :** Output of the data2tex aplication

```
OUTPUT -> muitos clientes do HOTEL-X classificaram o ...
   hotel como económico
(many clients of HOTEL-X classified hotel as cheap )
INPUT -> hotel-x hotel-service positive-eval ...
   many-comments economic
OUTPUT -> o estacionamento é acessível e disponível
(parking is accessible and available)
```

```
6 INPUT -> no-name parking positiv-eval half-comments ...
     accessible available
```

Human Evaluation of Summary

To test the results obtained in the project data2tex [85], a team consisting of 15 people were gathered. These 15 voluntaries had an age between 25 to 60 years, different professions and different levels of education but were all Portuguese native speakers. After several tests using different variations of the corpus used, the results were satisfactory, with only ten per cent of the phrases being rejected by the participants (the conclusion was that this probably happened because the data used to generate the sentences has not been big enough).

4.4 Graphical information for end-users

During the development of this dissertation, a paper was written [84] to show the importance of extracting useful information from the comments made to a hotel. The hotel management can later use this information to improve the hotel.

In the following charts developed for the paper (Fig. 4.6), we can see the comparison between two hotels (right chart) and the information about the comments made in two different years for a hotel(left image).

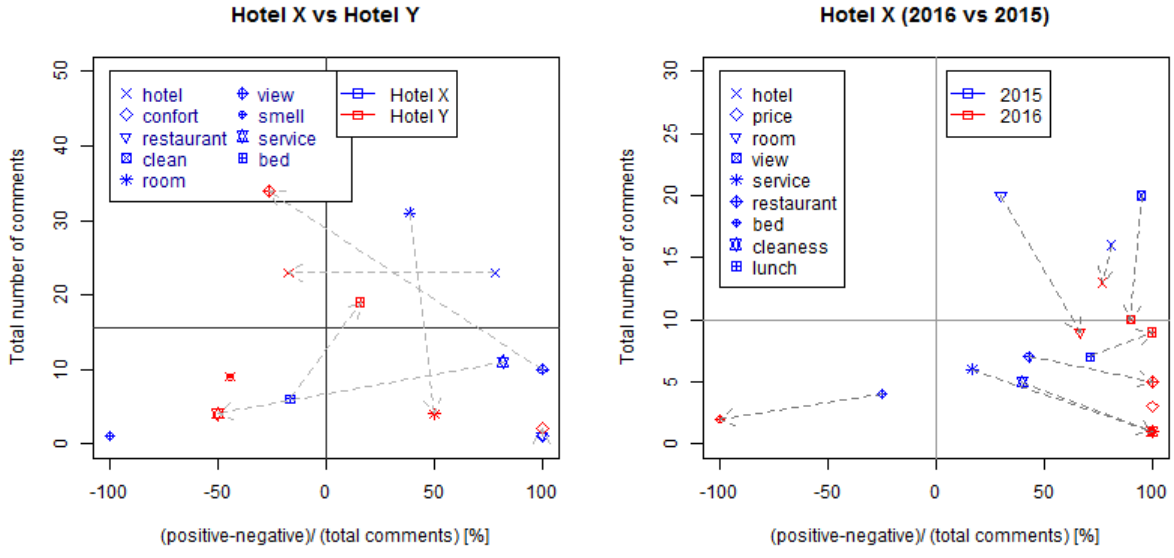


Figure 4.6: Example of results. At left, comparison between two hotels; at right, comparison of two different years for one hotel.

The objective of these charts is to help to condense all the information of hundreds or even thousands of comments, so it is easier to access the vital information. For example, in the following chart 4.7 (also developed for the paper), we can see the evolution of the accumulated positive minus negative evaluations for “hotel”, “rooms”, and “smell”.

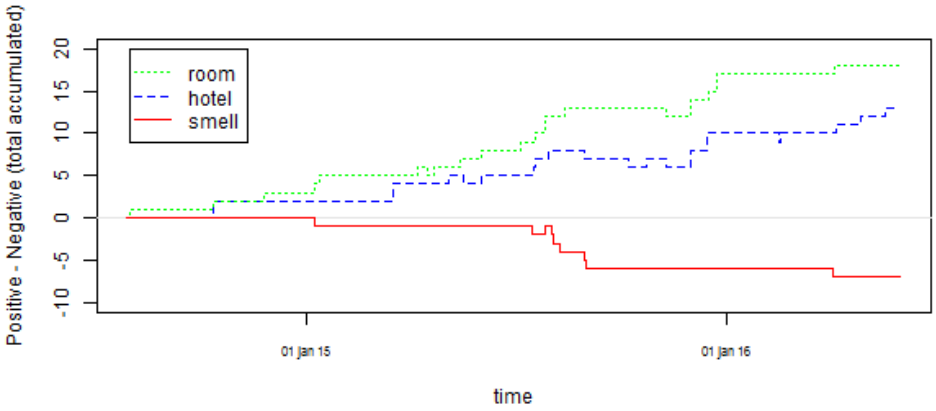


Figure 4.7: Example of evolution over time of a subset of the aspects of an hotel operation that our Information Extraction system considers.

4.5 Other Applications of the Pipeline

The NLP pipeline used in this dissertation was also used in project Voluntage4seniors [1] project. The structure of the solution developed for Altice lab in collaboration with the University o Aveiro is presented in Fig. 4.8.

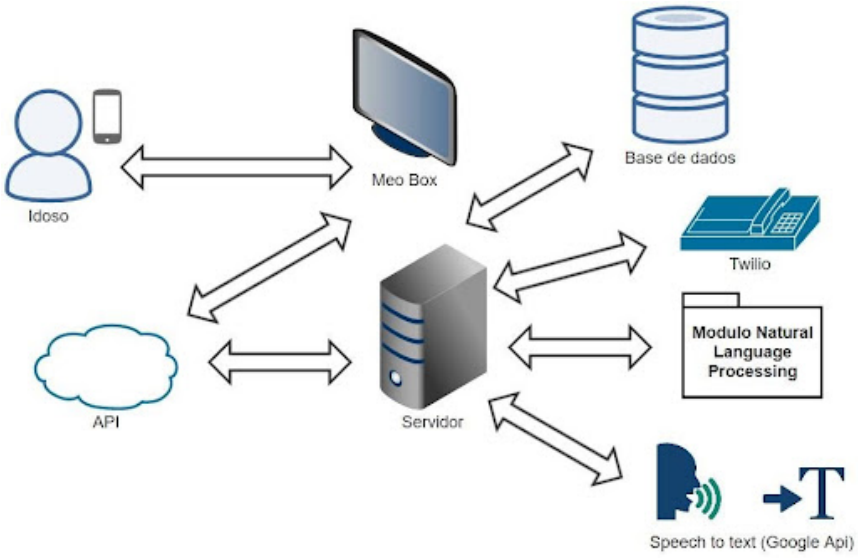


Figure 4.8: VoluntAge4Seniors Architecture

The main use of the pipeline in Voluntage4seniors was to transform requests (made by the elderly by phone) into simpler requests and to extract the date of the request.

In Voluntage4seniors, the pipeline used followed the same logic as the one created for the thesis. The two main difference was in the NER module 4.9 because the information that needed to be extracted was different, and in the information retrieval module, the raw data came in the form of voice to text.

Human Evaluation

During the development of the VoluntAge4Seniors project, at least four different tests with volunteers were performed. The first two were mainly to test the accessibility of the application (because all the volunteers were people 65+ years, the application should be easy to understand and navigate). The third test 4.10 4.11 4.12 made with twelve volunteers [2] was the most important to test the pipeline that was developed during this dissertation. In the pictures 4.10 4.11 4.12 we can see the elderly interacting with the

[1] Voluntage4seniors was a project created to help the elderly ask for help for simple tasks, such as going shopping, cooking, cleaning, etc.

[2] News coverage https://www.regiaodeaveiro.pt/pages/627?news_id=1720 and https://www.metronews.com.pt/2019/02/27/seniores-de-albergaria-a-velha-testam-novo-servico-tecnologico-desenvolvido-pela-universidade-de-aveiro/

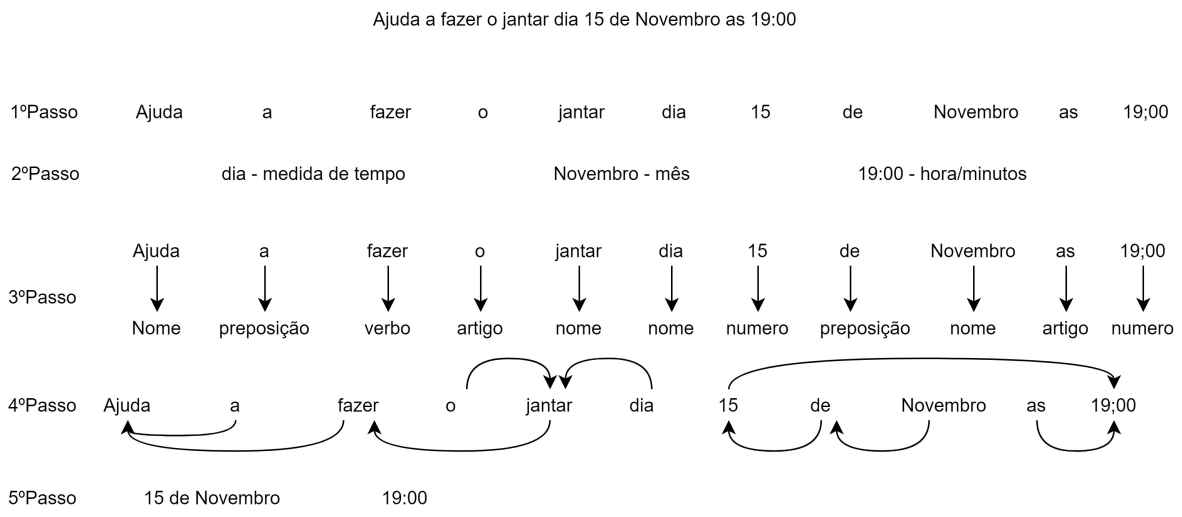


Figure 4.9: Figure showing how the hours were extracted in the project Voluntage4seniors

application. When the program well translated the phrases (the system had a voice to text module that depended on the volunteer's voice and did not translate how it should text), the date of the request was always extracted. In more than seventy per cent of the cases, a short description was also well extracted (sometimes the short description did not catch all of the vital information, so it was labelled as incorrect).

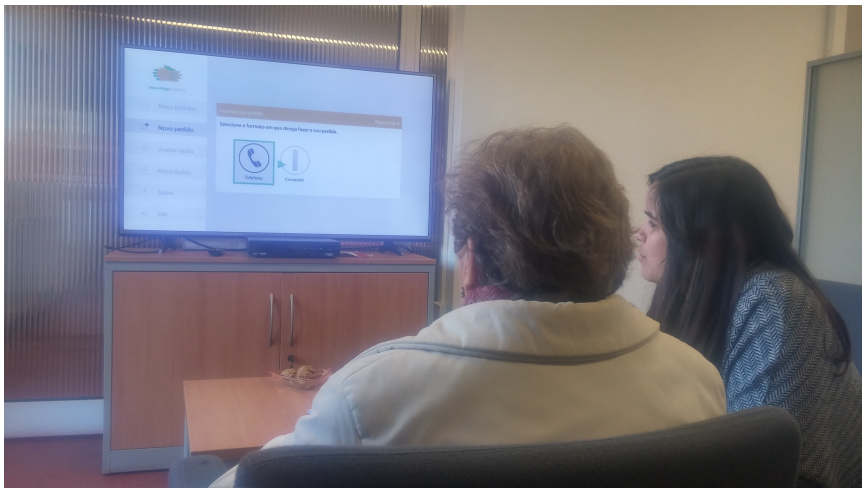


Figure 4.10: The user is listening to an explanation of how the application works.

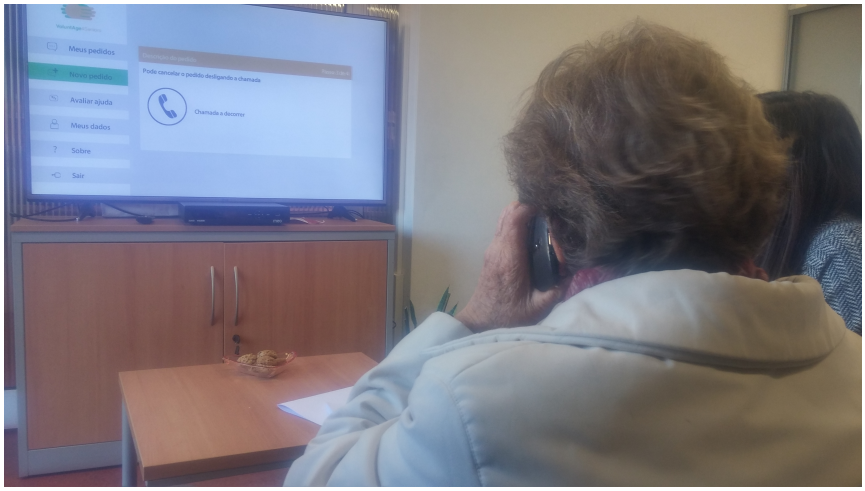


Figure 4.11: The user is making a request using the telephone.

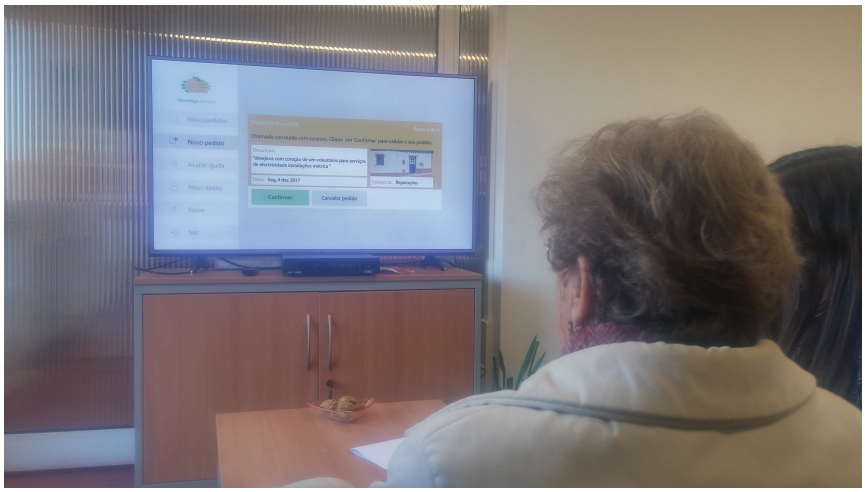


Figure 4.12: The user interacts with the application and sees the request that he did.

Chapter 5

Conclusion

5.1 Work summary

This dissertation had the following steps:

Learning - First of all, was learning what NLP meant and how could it have been implemented. Then was learning all what all the steps in the pipeline, what they do and how to develop them. After this, there was a needed to learn how to use a crawler to get the information from the websites, work with Apache Jena, and use and create an ontology.

Experiments with tools – During the development of this Dissertation, several tools have been experimented with. For example, the application Rembrandt was used for the NER block, and the Wikipedia API was also used to create a simple NER.

The Stanford NLP was used for the parser, and after doing some tests with the English language, it demonstrated promising results. Unfortunately, at the time, there was no good support for the Portuguese language. The only support for the Portuguese language is the LX parser based on an old version of the Stanford NLP and uses an old Portuguese corpus. So, in the end, the TreeTagger was used after doing some tests with the Portuguese corpus and having good results.

Mallparser at the time was one of the best dependency parsers that worked with the Portuguese language. Even so, some modifications had to be done to work with the TreeTagger (different label uses).

Protégé was also used to create and facilitate the use of an ontology.

Enter Tourism area an eWOM – Nowadays, several platforms gather and show information about hotels. Most of these platforms have information such as name, location, prices, availability and commentaries about those hotels. The main problem about those platforms is that they show much raw information that could have been displayed less cluttered.

For the hotel manager/owner, there are applications, for example, Booking or Tripadvisor, that have information about the score of the hotel (the higher the score, the more people liked the hotel). Although if there is a need to know what people like more or dislike about it, there is a need to sieve through all the commentaries to get the relevant information(sometimes we are talking about thousands of commentaries).

Acquiring Data – To get all the required information, two websites were studied (Booking and Tripadvisor.) Both add no support from an API, so it would be necessary to use a crawler. In the end, Booking was used because it added a lot more commentaries than Booking, and the information was easier to mine.

Pipeline Development – After experimenting with several modules that can be used in NLP, the pipeline that was used contained the crawler, a sentence splitter and a tokenizer, a POS tagger and lemmatization, a dependency parser, and a module to process the sentimental analysis.

Collaboration with Data2text – The data obtained and processed for this dissertation was also used as input for the project Data2text. A module was made to generate all data processed in an easy to use JSON file.

Applying it to other domains – The pipeline that was developed during this dissertation was also used in the project VoluntAge4Seniors. In this project, the module sentimental analyse was discarded, and the NER module was changed (the important information, in this case, was When and Where).

5.2 Main results and conclusions

Main results of the work:

- A first complete pipeline (when it was created in 2017) for eWOM on Hotels for Portuguese;
- The developed system is capable of providing new information to the Tourism stakeholders;
- The information extracted enabled the creation of and Data2Text prototype system;
- The pipeline proved helpful to other domains like VoluntAge4Seniors;

The main conclusions are:

- It is possible to create information extraction pipelines for Portuguese with available tools and resource, not off-the-shelf but with reasonable effort;
- The amount of valuable and synthesize information displayed by websites in the hotel industry still has space to grow;

5.3 Future Work

The work can be improved or continued in several aspects, including:

Improvement of the Dependency parser – Because people sometimes wrote reviews without verbs and very short (for example "Quarto sujo"), it was difficult for the Maltparser to construct relations between tokens in those sentences. So when the program tried to sentiment analyse the critical words (the named entities), it would fail because the Maltparser did not create the proper relations between the tokens. To resolve that problem, new rules would need to be created to sentiment analyse those short sentences.

Application frontend – A user-friendly application can be created for both the hotel manager and the possible clients of the hotels. These applications would be beneficial to show all the information that was processed by the application.

Access and process other eWOM sources – At this moment, Booking is the only source used to get information. Even if Booking has a good amount of reviews, other sources like Tripadvisor can enrich the amount of helpful information created by the application.

Evolve sentiment analysis and perform comparative evaluation – As new resources were created to perform sentiment analysis, an extensive test should be done to see if the source used is still the best one that can be used or if one of the new ones can get better results.

Better NE integration – Create a module to automatically add new import words to NER (a word that starts to appear with some regularity in the new comments and that can be added to the several lists already created).

Bibliography

[1] Brian Dean. *We alalyzed 5 million google search results. Here's What We Learned About Organic Click Through Rate*, 8 2019.

[2] James F Allen. Natural language processing. In *Encyclopedia of computer science*, pages 1218–1222. 2003.

[3] Toshiaki Nakazawa, Kun Yu, Daisuke Kawahara, and Sadao Kurohashi. Example-based machine translation based on deeper NLP. In *International Workshop on Spoken Language Translation (IWSLT) 2006*, 2006.

[4] Jiansong Zhang and Nora M El-Gohary. Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking. *Journal of Computing in Civil Engineering*, 30(2):04015014, 2016.

[5] Tarun Lalwani, Shashank Bhalotia, Ashish Pal, Vasundhara Rathod, and Shreya Bisen. Implementation of a chatbot system using AIand NLP. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST) Volume-6, Issue-3*, 2018.

[6] Oleksii Kuchaiev, Boris Ginsburg, Igor Gitman, Vitaly Lavrukhin, Jason Li, Huyen Nguyen, Carl Case, and Paulius Micikevicius. Mixed-precision training for NLP and speech recognition with openseq2seq. *arXiv preprint arXiv:1805.10387*, 2018.

[7] Monisha Kanakaraj and Ram Mohana Reddy Guddeti. Performance analysis of ensemble methods on Twitter sentiment analysis using NLP techniques. In *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, pages 169–170. IEEE, 2015.

[8] Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. Transformers:" the end of history" for NLP? *arXiv preprint arXiv:2105.00813*, 2021.

[9] W John Hutchins. The Georgetown-IBM experiment demonstrated in January 1954. In *Conference of the Association for Machine Translation in the Americas*, pages 102–114. Springer, 2004.

[10] Marc Moreno Lopez and Jugal Kalita. Deep learning applied to NLP. *arXiv preprint arXiv:1703.03091*, 2017.

[11] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[12] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[15] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[16] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[17] Jerry R Hobbs and Ellen Riloff. Information extraction. *Handbook of natural language processing*, 15:16, 2010.

[18] Johannes Starlinger, Madeleine Kittner, Oliver Blankenstein, and Ulf Leser. How to improve information extraction from German medical records. *It-Information Technology*, 59(4):171–179, 2017.

[19] Alexander Yates, Michele Banko, Matthew Broadhead, Michael J Cafarella, Oren Etzioni, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 25–26, 2007.

[20] Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. A survey on open information extraction. *arXiv preprint arXiv:1806.05599*, 2018.

[21] Yen-Pin Chen, Yuan-Hsun Lo, Feipei Lai, and Chien-Hua Huang. Disease concept-embedding based on the self-supervised method for medical information extraction from electronic health records and disease retrieval: Algorithm development and validation study. *Journal of Medical Internet Research*, 23(1):e25113, 2021.

[22] Michal Laclavík, Štefan Dlugolinskỳ, Martin Šeleng, Marcel Kvassay, Emil Gatial, Zoltán Balogh, and Ladislav Hluchỳ. Email analysis and information extraction for enterprise benefit. *Computing and informatics*, 30(1):57–87, 2011.

[23] Ying Sheng, Sandeep Tata, James B Wendt, Jing Xie, Qi Zhao, and Marc Najork. Anatomy of a privacy-safe large-scale information extraction system over email. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 734–743, 2018.

[24] Kiran Adnan and Rehan Akbar. An analytical study of information extraction from unstructured and multidimensional big data. *Journal of Big Data*, 6(1):1–38, 2019.

[25] University of Pennsylvania. Department of Linguistics. *Transformations and Discourse Analysis Projects*. Number v. 16-22 in Transformations and Discourse Analysis Projects. 1957.

[26] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January 1966.

[27] *ACL-Association for Computational Linguistics*, 1962.

[28] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[29] Shantanu Kumar. A survey of deep learning methods for relation extraction. *arXiv preprint arXiv:1705.03645*, 2017.

[30] Minh-Tien Nguyen, Dung Le, and Linh Le. Transformers-based information extraction with limited data for domain-specific business documents. *Engineering Applications of Artificial Intelligence*, 97:104100, 01 2021.

[31] Xi Yang, Jiang Bian, William R Hogan, and Yonghui Wu. Clinical concept extraction using transformers. *Journal of the American Medical Informatics Association*, 27(12):1935–1942, 10 2020.

[32] Swarnali Daw and Rohini Basak. Machine learning applications using waikato environment for knowledge analysis. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pages 346–351. IEEE, 2020.

[33] Stephen R Garner et al. Weka: The waikato environment for knowledge analysis. In *Proceedings of the New Zealand computer science research students conference*, volume 1995, pages 57–64, 1995.

[34] Hamish Cunningham. Gate, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254, 2002.

[35] Fei Wu and Daniel S Weld. Open information extraction using Wikipedia. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 118–127, 2010.

[36] Michael Schmitz, Stephen Soderland, Robert Bart, Oren Etzioni, et al. Open language learning for information extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 523–534, 2012.

[37] Bogdan Babych and Anthony Hartley. Improving machine translation quality with automatic named entity recognition. In *Proceedings of the 7th International EAMT workshop on MT and other language technology tools, Improving MT through other language technology tools, Resource and tools for building MT at EACL 2003*, 2003.

[38] Antonio Jimeno, Ernesto Jimenez-Ruiz, Vivian Lee, Sylvain Gaudan, Rafael Berlanga, and Dietrich Rebholz-Schuhmann. Assessment of disease named entity recognition on a corpus of annotated sentences. In *BMC bioinformatics*, volume 9, pages 1–10. BioMed Central, 2008.

[39] Petter Bae Brandtzaeg and Asbjørn Følstad. Chatbots: changing user needs and motivations. *Interactions*, 25(5):38–43, 2018.

[40] Cyril Joe Baby, Faizan Ayyub Khan, and JN Swathi. Home automation using iot and a chatbot using natural language processing. In *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–6. IEEE, 2017.

[41] Eiji Aramaki, Yasuhide Miura, Masatsugu Tonoike, Tomoko Ohkuma, Hiroshi Masuichi, and Kazuhiko Ohe. Text2table: Medical text summarization system based on named entity recognition and modality identification. In *Proceedings of the BioNLP 2009 Workshop*, pages 185–192, 2009.

[42] Monisha Kanakaraj and Ram Mohana Reddy Guddeti. Performance analysis of ensemble methods on Twitter sentiment analysis using NLP techniques. In *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, pages 169–170, 2015.

[43] Hannah Bast, Buchhold Björn, and Elmar Haussmann. Semantic search on text and knowledge bases. *Foundations and Trends in Information Retrieval*, 10(2-3):119–271, 2016.

[44] Lisa F Rau. Extracting company names from text. In *Proceedings the Seventh IEEE Conference on Artificial Intelligence Application*, pages 29–30. IEEE Computer Society, 1991.

[45] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[46] Andrei Mikheev, Marc Moens, and Claire Grover. Named Entity Recognition without Gazetteers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, 1999.

[47] Erik F Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.

[48] Diana Santos, Nuno Seco, Nuno Cardoso, and Rui Vilela. HAREM: An advanced NER evaluation contest for Portuguese. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May 2006. European Language Resources Association (ELRA).

[49] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[50] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*, 2019.

[51] Sumam Francis, Jordy Van Landeghem, and Marie-Francine Moens. Transfer learning for named entity recognition in financial and biomedical documents. *Information*, 10(8):248, 2019.

[52] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*, 2017.

[53] YaoSheng Yang, Meishan Zhang, Wenliang Chen, Wei Zhang, Haofen Wang, and Min Zhang. Adversarial learning for chinese ner from crowd annotations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[54] Meng Fang, Yuan Li, and Trevor Cohn. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383*, 2017.

[55] Jiabao Han and Hongzhi Wang. Transformer based network for open information extraction. *Engineering Applications of Artificial Intelligence*, 102:104262, 2021.

[56] Hongbin Ye, Ningyu Zhang, Shumin Deng, Mosha Chen, Chuanqi Tan, Fei Huang, and Huajun Chen. Contrastive triple extraction with generative transformer. *arXiv preprint arXiv:2009.06207*, 2020.

[57] Simone Magnolini, Valerio Piccioni, Vevake Balaraman, Marco Guerini, and Bernardo Magnini. How to use gazetteers for entity recognition with neural models. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 40–49, 2019.

[58] Khaled Shaalan. Rule-based approach in arabic natural language processing. *the International Journal on Information and Communication Technologies (IJICT)*, 3:11–, 06 2010.

[59] Antonio Toral and Rafael Munoz. A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia. In *Proceedings of the Workshop on NEW TEXT Wikis and blogs and other dynamic text sources*, 2006.

[60] Christopher Amato and Guy Shani. High-level reinforcement learning in strategy games. volume 1, pages 75–82, 01 2010.

[61] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[62] Sudha Morwal, Nusrat Jahan, and Deepti Chopra. Named entity recognition using hidden markov model (hmm). *International Journal on Natural Language Computing (IJNLC)*, 1(4):15–23, 2012.

[63] Koichi Takeuchi and Nigel Collier. Use of Support Vector Machines in Extended Named Entity Recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002.

[64] Asif Ekbal and Sivaji Bandyopadhyay. Named entity recognition using support vector machine: A language independent approach. 39, 01 2010.

[65] Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598, 2000.

[66] K. U. Senevirathne, N. S. Attanayake, A. W. M. H. Dhananjanie, W. A. S. U. Weragoda, A. Nugaliyadde, and S. Thelijjagoda. Conditional random fields based named entity recognition for sinhala. In *2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS)*, pages 302–307, 2015.

[67] Hyejin Cho and Hyunju Lee. Biomedical named entity recognition using deep neural networks with contextual information. *BMC bioinformatics*, 20(1):1–11, 2019.

[68] Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. Transfer learning for named-entity recognition with neural networks. *arXiv preprint arXiv:1705.06273*, 2017.

[69] John M Giorgi and Gary D Bader. Transfer learning for biomedical named entity recognition with neural networks. *Bioinformatics*, 34(23):4087–4094, 2018.

[70] Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 182–192, 2018.

[71] Mohammad Al-Smadi, Saad Al-Zboon, Yaser Jararweh, and Patrick Juola. Transfer Learning for Arabic Named Entity Recognition With Deep Neural Networks. *IEEE Access*, 8:37736–37745, 2020.

[72] Roald Eiselen, Martin Puttkammer, Justin Hocking, and Albertus Kruger. CTexTools 2. 2018.

[73] Lluís Padró and Evgeny Stanilovsky. Freeling 3.0: Towards wider multilinguality. In *LREC2012*, 2012.

[74] Olivier Galibert, Sophie Rosset, Cyril Grouin, Pierre Zweigenbaum, and Ludovic Quintard. Structured and extended named entity evaluation in automatic speech transcriptions. 01 2011.

[75] Jason Baldridge. The opennlp project. *URL: http://opennlp. apache. org/index. html,(accessed 2 February 2012)*, 1, 2005.

[76] Cláudia Freitas, Paulo Rocha, and Eckhard Bick. Floresta sintá (c) tica: bigger, thicker and easier. In *International Conference on Computational Processing of the Portuguese Language*, pages 216–219. Springer, 2008.

[77] Mário Rodrigues, Gonçalo Paiva Dias, and António Teixeira. Knowledge extraction from minutes of portuguese municipalities meetings, 2010.

[78] Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *LREC*, volume 6, pages 2216–2219, 2006.

[79] Nuno Cardoso. Rembrandt-a named-entity recognition framework. In *quot; In Nicoletta Calzolari; Khalid Choukri; Thierry Declerck; Mehmet U? ur Do? an; Bente Maegaard; Joseph Mariani; Jan Odijk; Stelios Piperidis (ed) Proceedings of the Eigth International Conference on Language Resources and Evaluation (LREC'12)(Istambul 23-25 de Maio de 2012; Maio de 2012)*, 2012.

[80] Paula Carvalho and Mário J Silva. Sentilex-pt: Principais características e potencialidades. *Oslo Studies in Language*, 7(1), 2015.

[81] Sillas Gonzaga. *lexiconPT: Lexicons for Portuguese Text Analysis*, 2017. R package version 0.1.0.

[82] Ayesha Ameen, Khaleel Ur Rahman Khan, and B Padmaja Rani. Reasoning in semantic web using Jena. *Computer Engineering and Intelligent Systems*, 5(4):39–47, 2014.

[83] John H Gennari, Mark A Musen, Ray W Fergerson, William E Grosso, Monica Crubézy, Henrik Eriksson, Natalya F Noy, and Samson W Tu. The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-computer studies*, 58(1):89–123, 2003.

[84] António Teixeira, Pedro Miguel, Mário Rodrigues, José Casimiro Pereira, and Marlene Amorim. From web to persons–providing useful information on hotels combining information extraction and natural language generation.

[85] José Casimiro Pereira. *Natural Language Generation in the Context of Multimodal Interaction in Portuguese*, 2017.