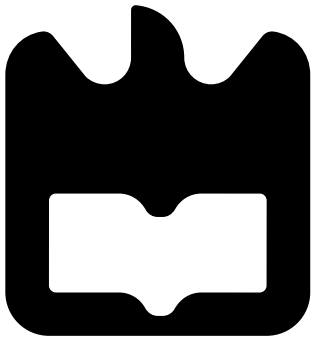
Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2018

Filipe Miguel da Costa Tavares

Análise de sentimento em comentários na web
Sentiment analysis in web comments



Filipe Miguel da Costa Tavares

Análise de sentimento em comentários na web
Sentiment analysis in web comments

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 de Doutor Sérgio Guilherme Aleixo de Matos, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor Joaquim Manuel Henriques de Sousa Pinto
Professor Auxiliar da Universidade de Aveiro (por delegação do Reitor da Universidade de Aveiro)

vogais / examiners committee

Doutora Carla Alexandra Teixeira Lopes
Professora Auxiliar, Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

Doutor Sérgio Guilherme Aleixo de Matos
Professor Auxiliar, Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (orientador)

agradecimentos

Queria aproveitar este momento para agradecer a todos os que me ajudaram, direta ou indiretamente, neste percurso académico.

Principalmente aos meus pais e irmã pelo apoio incessante e por toda a compreensão, nas fases boas e nas menos boas, independentemente de tudo. Pelo seu amor e por todos os ensinamentos diários, pois devo maioritariamente a eles ser quem sou. À minha namorada por me ter aturado, ajudado e sofrido com paciência ao longo destes anos todos (não deve ter sido fácil), com ela as coisas ficam mais fáceis. Ao meu afilhado e prima por todos os risos, brincadeiras e alegria. Aos meus avós pelas palavras de preocupação, de afeto e de estímulo e por serem como uns pais para mim. Na lhaneza das palavras e na longa e dura experiência de vida, reside o verdadeiro saber. Aos meus tios e madrinha por me terem sempre ajudado e por todo o auxílio que me prestaram e à restante familia, que sei que sempre que precisar estão lá para me estender a mão, ou até um braço.

Um especial agradecimento para todos os meus amigos, pelos risos, convívios, companheirismo e estima. Aos meus orientadores pela disponibilidade e conhecimento transmitido.

Por fim, um grande obrigado à Universidade de Aveiro pela formação de excelência! E obrigado a ti, por estares a ler esta tese.

Resumo

Hoje em dia milhões de pessoas usam a internet. Preferem consultar a Wikipedia em vez de procurar por informações numa enciclopédia offline, preferem um site de viagens ou um blog em vez de uma agência de viagens ou de panfletos. Este tipo de sites ajudou a melhorar a vida das pessoas. Os comentários online, por exemplo, têm o potencial de fornecer uma visão aos futuros compradores sobre os produtos em avaliação, como é o caso da sua qualidade. Além disso, estes comentários ajudam também os comerciantes a ter uma ideia do seu próprio produto. Esta tendência tem assim influência em muitas indústrias, como turismo ou culinária. No entanto, o verdadeiro valor dos dados da mídia social é raramente descoberto devido à sobrecarga de informação apresentada. Sendo que por vezes esta informação também não é facilmente acessível. Assim, foi considerado importante extrair este tipo de dados (comentários) e trabalhá-los de maneira a obter informações úteis. O objetivo desta tese prende-se então em desenvolver um sistema que obtenha e analise revisões sobre hotéis e restaurantes no TripAdvisor, em português, criando por fim uma aplicação web para visualização das informações obtidas. Em termos de análise, os objetivos foram divididos em dois: classificação multiclasse e binária dos comentários usando classificadores de aprendizagem automática e associar os parâmetros dos locais (por exemplo, 'quarto') a marcadores de opinião (por exemplo, 'ótimo'), dando-lhes uma pontuação (de um a cinco). Depois da obtenção e pré-processamento dos dados foi então feita a análise de sentimento destes. Para a primeira tarefa de análise foram utilizados quatro classificadores: Naive Bayes, Random Forest, Support Vector Machines (SVM) e uma rede neural LSTM, sendo a LSTM a que obteve melhores resultados (78% e 89% de exatidão para a classificação multiclasse e binária, respetivamente). A segunda tarefa foi completada com a ajuda da ferramenta *SyntaxNet* e de modelos de redes neurais, tendo sido encontradas e classificadas muitas associações.

Abstract

Nowadays, millions of people use the internet. They prefer to consult the Wikipedia instead of searching for information in an offline encyclopedia, prefer a travel website or a blog rather than a travel agency or a pamphlet. This type of site has helped improve people's lives. The online reviews, for example, have the potential to provide an insight to the future buyers about the product, such as its quality. In addition, these reviews also help the marketers getting an idea about their own product. This trend has influence in many industries, like tourism or culinary. However, the true value of social media data is infrequently discovered due to the overload of information. Furthermore, it is not easily accessible sometimes. So, extracting and processing these data to obtain useful information is important. The purpose of the work described in this thesis was to develop a system which collects and analyzes Portuguese reviews about *Tripadvisor* hotels plus restaurants and create a web application for visualization of the information obtained. In terms of analysis, the objectives were divided in two: Multi-class and binary classification of the comments using machine learning classifiers and associate the parameters of the places (e.g. 'bedroom') to feeling markers (e.g. 'great') giving them a punctuation (one to five). After obtaining and pre-processing the data the sentiment analysis followed. For the first task four classifiers were used: Naive Bayes, Random Forest, Support Vector Machines (SVM) and an LSTM neural network, being LSTM the one that obtained better results (78% and 89% of accuracy for multi-class and binary classification, respectively). The second one was completed with the help of the *SyntaxNet* tool and neuronal network models, with various associations found and classified.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The internet has been with us for approximately thirty-eight years [1]. But only in the last past couple of decades it has become a daily and universal source of information in many people's lives around the world. What is the impact of that?

Numerous people nowadays instead of consulting an offline encyclopedia use *Wikipedia*, prefer a travel website or a blog rather than a travel agency or an offline pamphlet [1]. It is easily accessible and faster for them accessing this information using only their fingers and a device than having to look at several places for the information that they have in one single place - the internet.

The growing interest in websites 2.0 and the development plus improvement of social media platforms like *Twitter* or *Facebook* and of social review sites like *TripAdvisor* have a lot of influence on several industries, such as tourism, hotels and culinary.

This influence is due to the expanding importance and number of customers online comments and, as such, many studies report that these reviews are regarded by many people (me included) as the best and most reliable sources of information, influencing the consumer purchasing process decisively [2]. In addition, these reviews also help the marketers getting an idea about their own product. As we can see in Figure 1.1, the average number of reviews that a typical business has on some of these platforms is increasing year after year.

So, the trend of online comments and their popularity led to the word-of-mouth and professional critics become increasingly outdated, nowadays people trust more on casual consumers like themselves (it is like a personal recommendation).

TripAdvisor, for example, is the most popular travel reviews website in the world and that makes it an important tool for trip planning. Every year the number of reviews for attraction raise, as it is possible to verify in Figure 1.2. Currently, it has over 630 million reviews/ opinions and 455 million average monthly unique visitors [3].

[1] There is no consensus on the exact date on which the modern internet emerged, but it was sometime in the mid-1980s.

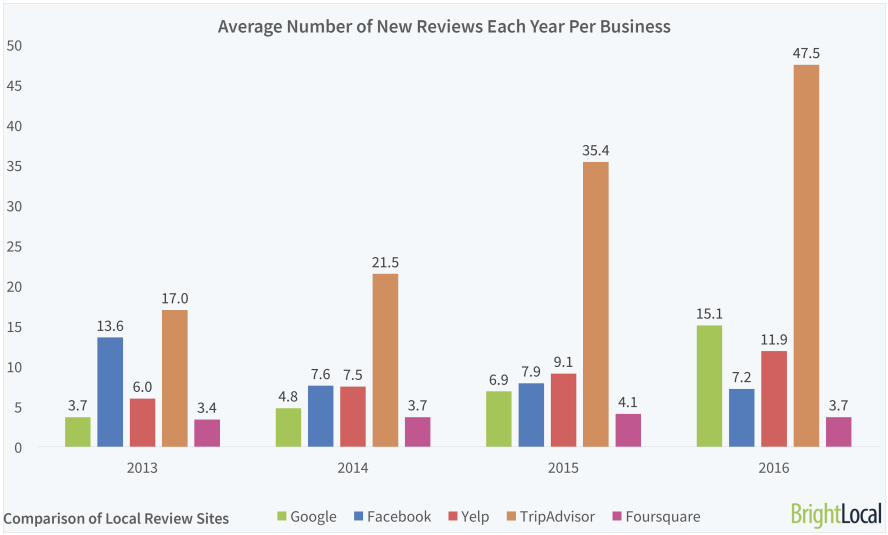


Figure 1.1: Average number of new reviews gained in each year for a business using some web platforms.
Available at `www.brightlocal.com/2018/01/17/comparison-of-local-review-sites`. Accessed on 29 May, 2018.

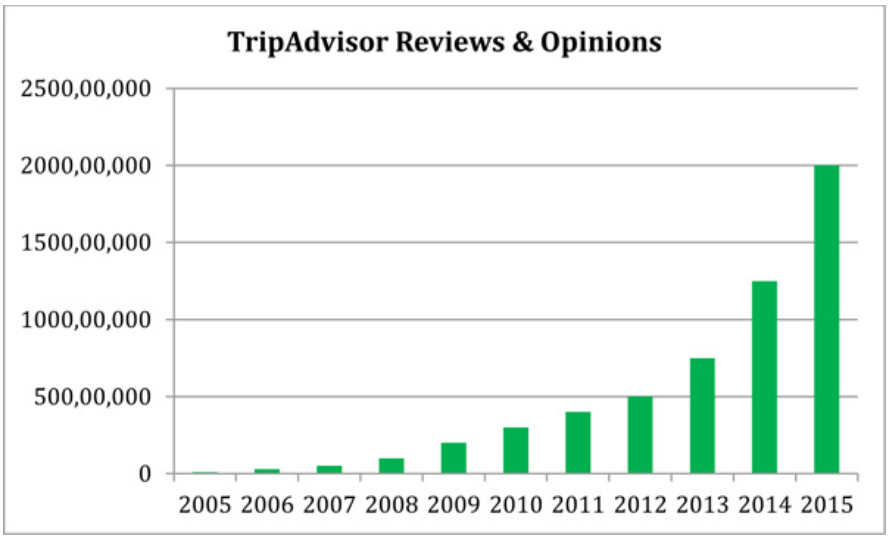


Figure 1.2: *TripAdvisor* reviews and opinions from 2005 to 2015.
Available at `www.travelmail.in/tripadvisor-celebrates-15-years`. Accessed on 29 May, 2018.

In addition to the growing number of comments, the economical crisis also played a key role in this boom. People are now more careful in how they spend their money and if the attraction is good enough to spend cash on it. Before they go eating in a restaurant or sleeping in a hotel, they do not just check out the best ones in the area they want to visit, instead, they examine which ones are fittest for the price they want to spend.

Despite this trend, the true value of social media data is rarely discovered due to overloaded information. Furthermore, it is not easily accessible sometimes. So, it is important to extract this data and work on them to obtain useful information for people in general.

The purpose of the work described in this thesis was to develop a system which collects and analyzes Portuguese reviews about *Tripadvisor's* hotels plus restaurants and create a web application for research/ visualization of the information obtained. In terms of review analysis, the objectives will be divided in two:

- Classify comments using machine learning classifiers.
- associate parameters of the attractions (e.g. bedroom, food, service) to feeling markers (e.g. great, fantastic, bad) and give them a punctuation (from one to five), using lexical resources;

The associations obtained for each review will be indexed and viewable at a web interface.

We use the *scrapy* tool to obtain the comments. Then, their classification was done by four different classifiers: Support Vector Machine, Naive Bayes, and Random Forest. The identification plus classification of the associations found were done using some *Python* tools like *SyntaxNet* and *Textblob*. Finally, this information was saved using the *elasticsearch* framework and the web interface that explores it was created with *Django*.

Chapter 2 explores the sentiment analysis in datasets obtained from web and its state-of-the-art techniques, including the methods that researchers have developed to meet this challenge. **Chapter 3** discusses how the data used was obtained and what kind of pre-processing it suffers before it is used for sentiment analysis. **Chapter 4** specifics which machine learning methods and frameworks where used to classify the reviews and to obtain the associations contained within each comment. **Chapter 5** takes a close look into the results obtained from previous chapter. Finally, **Chapter 6** presents a brief conclusion and some suggestions regarding what may be added to this project in the future.

Chapter 2

State of the Art

Research is what I'm doing when I don't know what I'm doing[1].

This chapter is an introduction to sentiment analysis in web comments, done through classification and opinion mining in data obtained from websites.

As said before, nowadays forums, review sites and blogs (e.g., *TripAdvisor.com* and *Yelp.com*) have been gaining more and more popularity and this leads to an enormous heap of data, namely in the form of user reviews, opinions, ratings, and arguments about different things, like products, hotels, food, brands, politics, among other things. These opinions greatly influence the other users, readers, buyers and product vendors.

Many people before buying a product, staying in a hotel or eating the food of a certain restaurant, check out this type of sites to see if their choices are really the best (by looking at user comments and/or their evaluations).

Social networks, like *Twitter* or *Facebook*, are also a place were people often go to check other user ideas and share opinions and information (like before/after travelling or buying something) [4]. This word-of-mouth has an important impact on the product or location success (92% of consumers believes in their friends recommendation and 20% to 50% of purchases just happen because of that word-of-mouth [5]).

Many of the review sites, however, have not easily accessible useful information, and consumers in their search want to find practical information as quickly as possible.

The data that is only accessible through a textual evaluation will lead to the user frustration and will also be very time-consuming for them.

The *TripAdvisor* site, for example, does not have any information about what aspects are presented in a user review and what are the feelings about them or what are the best and worst aspects of a particular hotel.

So, because of the growing interest in these Web 2.0 websites and in order to assist them, it is necessary and interesting that the unstructured data presented in these sites is extracted, analyzed and structured. Then, these data, through machine learning or/and opinion mining, can be used to achieve very useful information.

For this to happen the first question that need to be asked is: What are datasets and how can they be obtained?

[1] Wernher von Braun

2.1 Datasets

A dataset is defined as a data collection. As a general concept, it refers to the fact that some existing data compilation is represented or coded in some form suitable for better usage or processing. These random data are the raw material for information, and information is the raw material for knowledge [6]. Thus, in general, we can assume that everything has potential to be a dataset.

As such, in order to obtain the desired results, we need to be sure that the datasets obtained are clear and suitable for the tasks. These collected set of values are very important for the accuracy and relevancy of any prediction system, classification and machine learning model. So, for example, bad datasets will lead to bad classification systems.

Nowadays one way to obtain these type of datasets can be through a few online repositories of large amounts of data for machine learning projects. People can access them easily and get these sets of data. These sites also cleaned up the datasets beforehand (messy data is bad), and allow for testing of algorithms very quickly. Table 2.1 shows us some of them.

Site	Link	Description
Kaggle	https://www.kaggle.com	Data science community that hosts machine learning competitions. It is necessary to sign up and accept their terms before downloading any data.
UCI Machine Learning Repository	https://archive.ics.uci.edu/ml/index.php	One of the oldest sources of data sets on the web. The vast majority of datasets are clean and ready to be applied on machine learning methods
Webhose.io	https://webhose.io/datasets	Datasets can be accessed and analyzed by anyone. They are splited by language, category, reviews, etc.
Yelp	https://www.yelp.com/dataset	The Yelp dataset is a subset of their businesses, reviews, and user data. Available in both JSON and SQL files. These datasets are only for use in personal, educational, and academic areas.
Quandl	https://www.quandl.com/	Quandl is a repository of economic and financial data. Many datasets require purchase, just a few are free.

Table 2.1: Online repositories for Machine Learning Projects

2.1.1 Data extraction

Another way of obtaining datasets from Web 2.0 is extracting them manually or by using web data extraction tools (web scrapers and crawlers).

Manual or automated extraction?

But which of these two methods is most reliable? Manual entry or automated extraction?

The first option includes the traditional and familiar copy and past process, which is an inefficient and hectic task for a significant amount of data because it requires a lot of time or/and people. This approach might be viable if it is only required a tiny amount of data. Others drawbacks of manual entry are the ambiguity and human errors. Source documents can be unclear, with unlabeled or missing fields. This leaves data entry clerks to make their best guess as to what should be entered. The enormous amount of data that leads to failures, caused by fatigue or distraction, occurs more often.

So, automated extraction is more reliable, this process provides an auditable workflow of information (high speed extraction), something that the first option can not give. These type of extraction also allows people who previously entered data to do more meaningful work.

Although automated extraction is quite superior to manual, it can also produce errors. No process is 100 percent error-free and this extraction also does not need to be - it just needs to be better than the manual process that it is replacing.

In summary form, the most basic type of collection is the manual download of the pages (copying and pasting the content) and this can be done by anyone. But this is a very time-consuming technique, and because of that it is normally used through software that simulates a human website navigation to extract specific information automatically.

Frameworks to automated extraction

The most common keywords associated to these automated extraction are the web scraping and crawling terms. Now the questions are "what is web crawling and scraping? And what are the biggest differences between them?".

- Web crawling (also known as web indexing) is used to automatically download any type of information from one or more web pages and index it into a database, using for that bots which are also famous as crawlers. This type of extraction is used by major search engines, like *Google* and *Yahoo!*. For example, if we need all the phone numbers of a website, we should crawl the pages of that specific website and match the information crawled and indexed from them with phone numbers regular expression to find all of these numbers.

- Web scraping is a computing technique used to extract large amounts of data from the web (more focused and intentional than web crawling). Briefly speaking, a computer program visits one or more websites and extracts the required data, all within a tiny fraction of time (compared with personal extraction). This program interacts with the websites by mimicking a human visitor. But instead of displaying on a screen the page served by the site, the web scraping tool will save the required data to a database or a local file. All of this information can be seen at Figure 2.1.

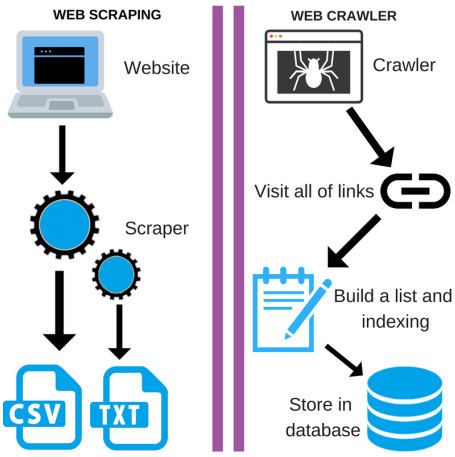


Figure 2.1: Web Scraping Vs Web Crawling

Automated extraction research

There are many projects that used these two techniques. Rafael Oliveira and Renata Porto, for example, used a web scraper tool to collect data from *TripAdvisor*. They filtered the scraping for relevant information related with Minas Gerais' tourist attractions. The data extraction tool used is known as *import.io*[2]. With these datasets they are able to extract specific information, like satisfaction levels of visitors and, for each attraction, the period of most visits. The extracted information will be able to assist the state authorities and municipalities [7].

Maleerat Sodani collected the data (related to hotels) from *TripAdvisor* and *Agoda* using the tool *import.io* and then he analyzed these reviews and developed a taxonomy of features for them. After this, he gave these features as input to different classifiers in order to obtain a classification for each hotel with a probability associated that indicates how accurately these evaluations can be predicted [11].

Madelyne Velasco et al. collect data from *TripAdvisor* website related to Quito city in order to describe this city based on geo an lexical information included in the user's reviews. To get these reviews given by users who visit or live in Quito city, they used Python[3] scripts combined with the *scrapy* library[4] [8]. This library is an open source and collaborative framework that is used for fast and simple data extraction from websites [9].

Eva Sánchez-Amboage et al. also used the *scrapy* library to obtain commentaries related with restaurants located in the most representative cities of Ecuador, like the Coast Region, Highlands, Amazon Region and Galapagos Islands. These datasets (commentaries) were used to know/analyze the travelers profiles and their preferences to obtain the reputation of these restaurants and establish a comparison between the most valued indicators of each one [10].

Francisco Villarroel-Ordenes et al. collected reviews from different online customer review sites (*Amazon.com*, *Bn.com*, *tripadvisor.com*), using for that *Monzenda* [5]. Then, in general,

[2]A free web site scraping tool that runs online with no need to software installation on a computer - https://www.import.io/

[3]https://www.python.org/

[4]https://scrapy.org/

[5]Another web scraping tool - https://www.mozenda.com/

with this dataset they analyzed the sentiment expressed by users about two different themes, books and hotels [12].

After obtaining these appropriated and wanted datasets, it is necessary work on them in order to extract useful information. This information then needs to be analyzed. For this purpose, opinion mining and sentiment analysis are very interesting tools. Their analysis can be used to develop applications that can assist people in making purchase decisions and/or help business organizations to determine the strengths and weaknesses of their products, for example.

A basic task in sentiment analysis is classifying the polarity of a given sentence from a document or review, using for that machine learning methods (classifiers) that have as output a positive or negative score.

2.2 Classification

Sentiment analysis refers to the management of sentiments, opinions, and subjective text. It provides the comprehension of the information related to public views, as it analyzes different public sentences and reviews. It is a verified tool for the prediction of many significant events such as box office performance of movies [13].

These public reviews can be used to evaluate a certain entity, i.e., person, product or location, and might be found on different websites like *Amazon* and *TripAdvisor*. The evaluation occurs by categorizing their opinions, for example, into negative, positive or neutral.

So, the purpose of sentiment analysis is to automatically determine the expressive direction of user reviews. There are 2 main techniques for Sentiment Analysis: lexicon based (this one will be described at Section 3.) and machine learning based techniques [13].

These days the term Machine Learning is being used by every company in every industry. They're the type of words that make any ahead company CTO mouth-watering. But what is Machine Learning?

Machine Learning is a field of computer science that gives computers the ability to learn without being explicitly programmed, it explores the study and construction of algorithms that can learn from data and make predictions on it [21].

So, given one or more inputs to a classification[6] model will lead them to predict the value of one or more outcomes. For example, they will be able to classify if an email is "spam" or "not spam" if they are fed beforehand with a large set of emails, labeled or not.

There are three approaches to machine learning, which differs in the way of predict the results: supervised, semi-supervised and unsupervised. In a supervised model, the training dataset that is fed into the classification algorithm is labeled. It means that it is marked with some type of classification, like "spam" or "not spam".

After training with these type of data, the model will work on the test data sample and compare them with the training set, to determine if there is, for example, a "spam" email. This type of learning is called "Classification".

Semi-supervised models are a class of supervised learning tasks but they also use techniques working on a small unlabeled dataset for training.

Unsupervised models on the other hand, are fed with an unlabeled dataset and look for data point clusters. They can be used to search data for similarities, detect patterns, and/or identify anomalies within a set of data.

[6]The output variables take class labels.

A typical use case for these models would be for finding similar images. This type of learning falls under "Clustering". Contrary to what happens in the other two models, in an unsupervised model there is no evaluation of the accuracy of the output (this happens because the examples given to the learner are unlabeled).

Do not forget that, in order to be able to achieve positive outcome it is necessary that the machine learning methods are fed with enough proper training data.

2.2.1 Naive Bayes

One of the oldest machine learning methods is the one based on the Naive Bayes technique and is still widely used in present days (such as Yoon Kim, Wang and Manning [29] [30], Alec Go, Richa Bhayani and Lei Huang [28] and some teams from SemEval-2017 [32]).

This technique has been studied extensively since the 1950s and its theorem states the following relationship equation [40]:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)} \tag{2.1}$$

Where y means a given class variable and the values between x_1 and x_n serve as a dependent feature vector.

It is a popular (baseline) method for text categorization, classification of documents as belonging to one category or to another (such as spam or not spam , etc.) using features based on word frequencies (bag-of-words models).

In machine learning, Naive Bayes classifiers belongs to the family of simple probabilistic classifiers and are based on the so-called Bayesian theorem[7] [40]. They work very well and often outperform more advanced classification methods when the inputs dimensionality is high [41].

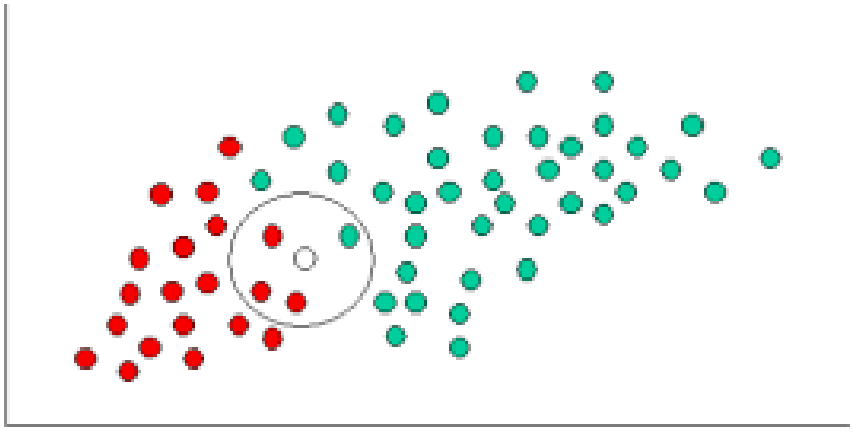


Figure 2.2: Chart composed by a dataset with GREEN and RED points.
Available at `http://www.statsoft.com/textbook/naive-bayes-classifier`. Accessed on 27 January, 2018.

[7]The probability of an event occurs, based on knowledge of prior conditions/events that are related to this event. For example, if a disease is related to the female gender, a person gender can be used to obtain a more accurately probability that she has the disease, using for this the Bayes theorem.

In order to see how this method works, we can attempt at the Figure 2.2. When a new point is inserted at the chart (represent by the white point) it is necessary to decide automatically which color it will have, naive bayes algorithm predict its color by calculating the prior probability of being green ($\frac{40}{60}$) and being red ($\frac{20}{60}$).

Based on the principle that the objects are well clustered, it is reasonable to assume that the more green or red points exists in the surroundings of the white point, the more likely is that this point belongs to that particular color. To measure this, a circle around the white point is draw (whose radius is chosen beforehand). In this case, 3 of the points inside these circle are red and just one is green. So the probability of being red and green will be $\frac{3}{20}$ and $\frac{1}{40}$, respectively [41].

Finally, in order to discover the final highest probability, that is the output of the Bayesian classification, is produced a combining of both of the previous probabilities. This is:

- Probability of white point being red: $\frac{2}{6} * \frac{3}{20} = \frac{1}{20}$
- Probability of white point being green: $\frac{4}{6} * \frac{1}{40} = \frac{1}{60}$

So, the prediction made by the Naive Bayes Classifier for the color of this point will be the red one.

2.2.2 Random Forest

The machine algorithm described above still is very important and should be looked into for someone who is a machine learning beginner. But there is another which also deserves special attention: Random Forests.

Random forests were introduced by Tin Kam Ho [42]. These machine learning models are also very popular and attractive thanks to their high execution speed.

They are an ensemble leaning method used for classification, regression [8], among others, that shows excellent performance even when most predictive variables are noisy, can be used when the number of variables is much larger than the number of observations and in problems involving more than two classes, and returns measures of variable importance [43]. They can also handle missing values, maintain accuracy for missing data and they will not overfit the model.

This type of machine learning method operates by constructing a large collection of decorrelated decision trees at training time and each one of them outputs a prediction class (after the train). The final class chosen by this algorithm is the most voted trough the decision trees.

Decision trees are very intuitive, even for beginners in the field of decision analysis. They are composed by four elements: decision nodes (represented by a square), uncertainty nodes (represented by a circle) that are associated with a probability of an event occurs, the payoff (measure of satisfaction) and the end nodes (represented by a triangle).

The optimal decision gave by the decision tree is obtained by calculating the rolled back expected value at each chance node in the tree and choosing the alternative with the highest value ate each decision node [44].

For example, in Figure 2.3, if the value associated to the work node is higher then the value of Go-Fishing node, the first node is the chosen one.

[8] Continuous values for the output variables.

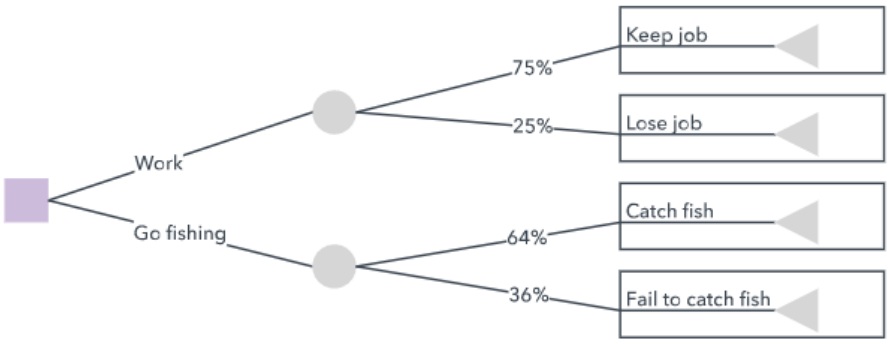


Figure 2.3: Decision tree, Available at `https://www.lucidchart.com/pages/decision-tree`. Accessed on 28 January, 2018.

2.2.3 Support Vector Machines

The deep learning methods[9] (the tendency in these days) are not necessarily better than traditional machine learning methods (like Naive Bayes, SVM, among others). Their achievement not only depends on an appropriate network structure, but also on the training dataset size. Neural networks usually are better for more training data.

Before this deep learning trend, one of the most popular methods was SVM. Support Vector Machines were firstly introduced by Vapnik and coworkers and are a class of supervised learning algorithms [22] [23].

Given a k kernel and an annotated training dataset, the SVMs discriminate between positive and negative examples by learning a linear decision boundary in a feature space defined by the kernel. This linear decision boundary is represented by a normal vector v which maximizes the margin between the positive and negative classes, as we can see in Figure 2.4. New unannotated examples are then mapped into this feature space and then predicted to be negative or positive based on which side of the gap their image fall [24].

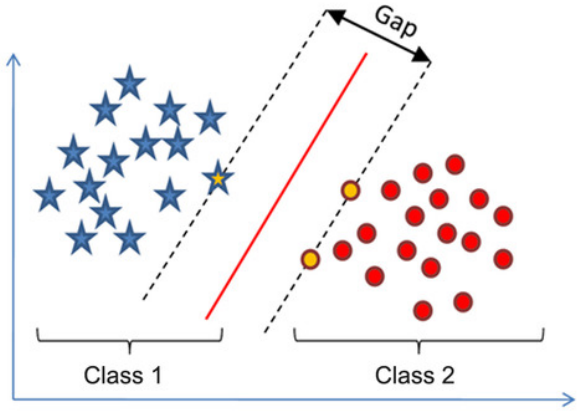


Figure 2.4: SVM linear classifier. Available at `https://www.slideshare.net/ankitksharma/svm-37753690`. Accessed on 26 January, 2018.

[9]It will be addressed in the next sub chapter.

What kernels do is to change the definition of the dot product in a set of mathematical functions that SVM uses to obtain the hyperplane. The kernel parameters define the type of hyperplane that is used to separate the examples per class. So, to a better adjustment to the training datasets, there are different kernel types.

There are four of them: *linear*, *polynomial*, *radial basis function (RBF)*, and *sigmoid*.

A Linear SVM kernel applies a linear hyperplane, instead the other kernels imply nonlinear hyperplanes. A more accurate difference between the first three kernels types can be seen at Figures 2.5, 2.6 and 2.7. Now that the visual explanation is done let's see, in a general way, how this happens.

A kernel is actually a function that maps the data to a higher dimension where this data can be separable. The data that is not linearly separable in 2-dimension will be projected up by some number of dimensions using a mapping function. In that dimension plane, the data will then be separable. Finally, the hyperplane can be projected back down again to get a decision boundary that will be able to separate the example points.

In a very simple way, this is what an SVM kernel does, and different kernels have a different algorithm that does this contrastively.

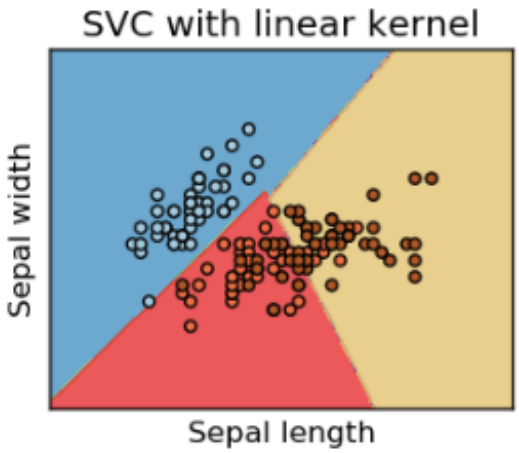


Figure 2.5: Support vector machine with 'linear' kernel. Available at `https://medium.com/@mohtedibf/in-depth-parameter-tuning-for-svc-758215394769`.
Accessed on 26 May, 2018.

In this example, the *linear*, *rbf* and *polynomial* kernels work equally well. When this happens the *linear* should be the chosen one, due to the higher complexity and expensiveness of the other two algorithms. But sometimes, the *linear* kernel just can not fit the dataset and an *rbf* or *polynomial* kernel should be applied.

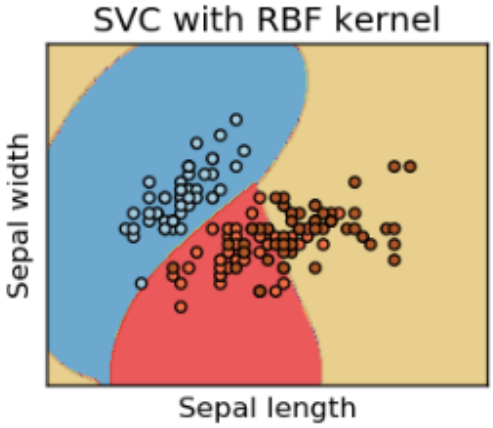


Figure 2.6: Support vector machine with 'rbf' kernel. Available at `https://medium.com/@mohtedibf/in-depth-parameter-tuning-for-svc-758215394769`. Accessed on 26 May, 2018.

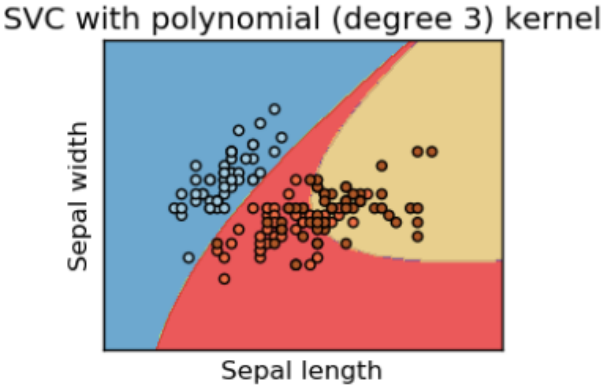


Figure 2.7: Support vector machine with 'polynomial' kernel. Available at `https://medium.com/@mohtedibf/in-depth-parameter-tuning-for-svc-758215394769`. Accessed on 26 May, 2018.

Apart from these three, there is still another kind of kernel, *sigmoid*. The function of this kernel only returns values between 0 and 1. Therefore it is more suitable for binary classification problems.

In addition to the kernel, there are two important parameters that need to be tuned, and so they were: *Gama* and *C*.

Gamma is a parameter used for non linear hyperplanes. This parameter determines how far the influence of a single training example reaches. In one hand, when a low value is set its influence is far. Contrarily, a high value means close influence.

Because of that the behavior of a model (like SVM) is very sensitive to the *gamma* parameter. If *gamma* is too large will cause overfit[10], if is too small, the model is very constrained

[10]Occurs when a model fits well the previously trained data but is ineffective in predicting new results.

and cannot capture the complexity or "shape" of the data.

As you can see at Figure 2.8, the increasing *gamma* leads to an overfitting as the model tries to perfectly fit the training data.

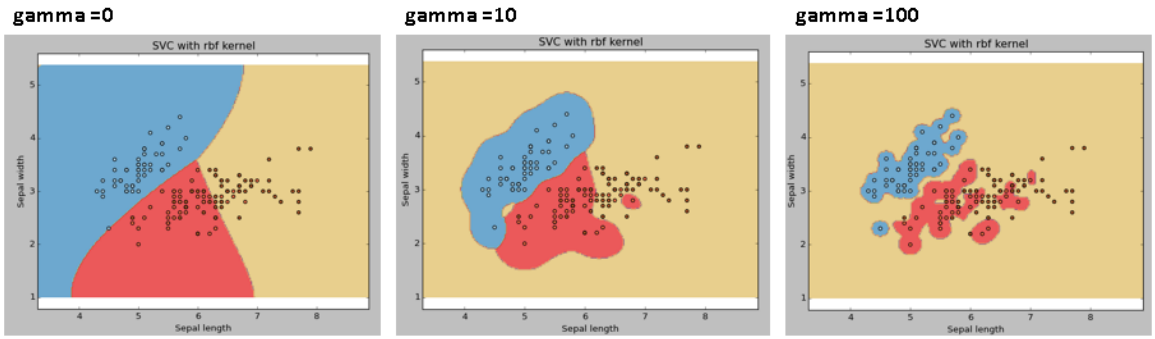


Figure 2.8: Contrast in results for several values of gamma. Available at `www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/`. Accessed on 26 May, 2018.

C is the penalty parameter of the error term. Its function is to control the trade-off between smooth decision boundary and the correct classification of the training examples (figure points).

As shown in the Figure 2.9, a lower value of c leads to a larger margin that separates hyperplane from samples, even if that hyperplane miss-classifies more points, and for a higher value the algorithm will only choose a smaller margin hyperplane if that does a better job of getting all the training examples classified accurately for each class.

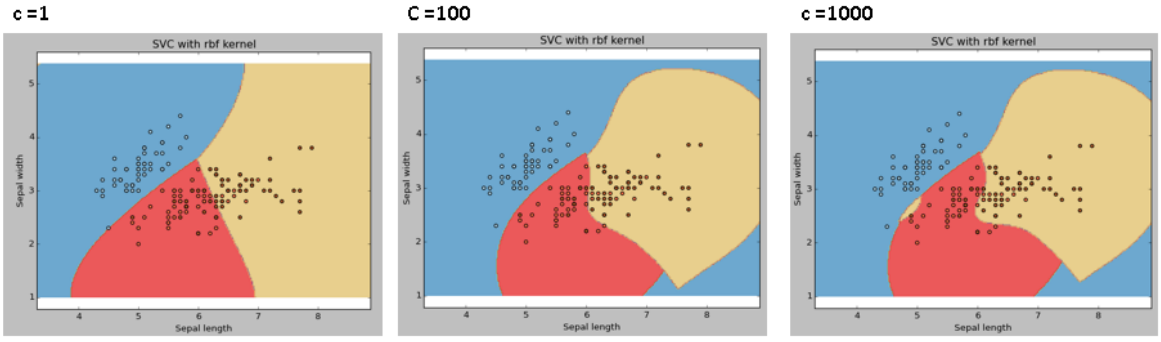


Figure 2.9: Contrast in results for several c values. Available at `www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/`. Accessed on 26 May, 2018.

SVM are still be very popular, firstly, due to its low prediction time. This low time occurs because the new test instances are assigned based just on their distances from the support vectors. Secondly, it has a regularization parameter, which makes the user think about avoiding over-fitting. Finally, it uses the kernel methods, so the users can build in expert knowledge about the problem via engineering the kernel [25].

2.2.4 Deep learning

In the actual days people from all over the world compete to produce the best machine learning models. One of the most popular models used nowadays is based on deep learning. After a brief search for this term in Google Scholar were obtained (on 15 January 2018) around 4 280 000 results since always, 461 000 results between 1999 and 2008 and 1 300 000 results from 2009 to 2018. As we can see, in the last 9 years there are about three times more projects using deep learning (deep learning is the thing actually). But what is deep learning?

Deep learning was firstly proposed by G.E. Hinton in 2006 and is part of a Machine Learning process which refers to Deep Neural Network [14]. They are shaped considering the biological nervous system and because of that contain numerous neurons in their huge network.

In other words, deep learning is based on a set of algorithms that attempt to model high level data abstractions using a deep graph with several layers of processing (composed of several linear and nonlinear transformations) [15][16], as we can see in the Figure 2.10. No matter what are the set of algorithms (functions - f(x)) used,a neural network is guaranteed for every possible input, x, the value of function f(x) will be always output from the network.

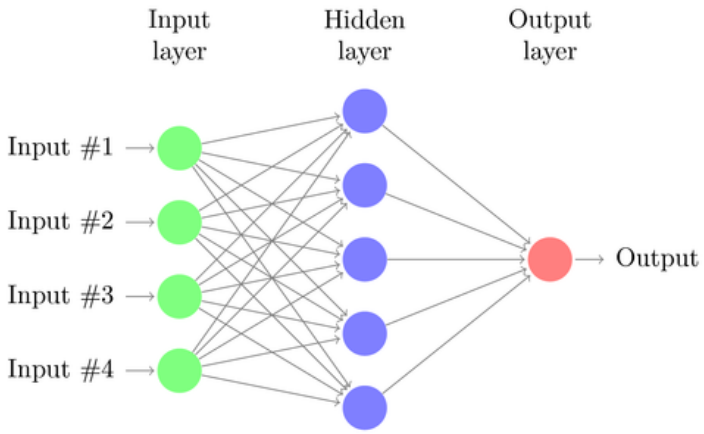


Figure 2.10: Basic Neural Network.
Available at `http://www.texample.net/tikz/examples/neural-network/`. Accessed on 27 January, 2018.

Neural networks are used for several applications, like text generation, vector representation, word representation estimation, sentence classification, sentence modeling and feature presentation and sentiment analysis [17].

Although there is always a neural network, there are many types of them that work differently and have different structures, such as CNN (Convolutional Neural Networks), RNN (Recurrent Neural Networks), Recursive Neural Networks (RNN), DBN (Deep Belief Networks) and more.

Comparing to fully connected networks, the CNNs need many fewer parameters with the same number of hidden units, which makes them much easier to be trained.

RNNs have gained massive attention in the field of NLP. Because of that, they have been employed to handle many tasks, like machine translation. They exhibit dynamic temporal behavior due to a directed cycle, which is formed by connections between their units [18]. A

RNN, using their internal memory, can process sequences of arbitrary length inputs.

The Long Short-Term Memory networks (LSTM) are recurrent neural networks (RNNs) that are trained using back-propagation trough time and were developed to take care of the vanishing gradient problem[11]. As such, they can be used to resolve difficult machine learning sequence problems and obtain state-of-the-art results (by creating large recurrent networks) [19].

LSTMs, instead of neurons, have memory blocks connected through layers. These memory blocks make this network smarter than a classical one and with a memory for recent segments (in another words, LSTM is a model for the short-term memory which can last for a long period of time) [19]. Because of this they are widely used nowadays for classifying texts, where it is necessary memory existence to identify dependencies in the text. So, let's understand a little more about this type of artificial neural networks.

All the people of the world understand each new word based on their understanding of previously learned words. They simply can not throw away their information and start thinking from scratch to understand a unknown word. Like people, to classify a text sequence its necessary some form of information persistence that allows to detect the dependency between different words. Traditional neural networks can not do that, but recurrent neural networks resolve this issue [20].

They are networks with loops in them, allowing the persisting of information and the detection of dependencies between terms. The figure 2.11 shows how RNNs allows information to be passed from one step of the network to this successor [20].

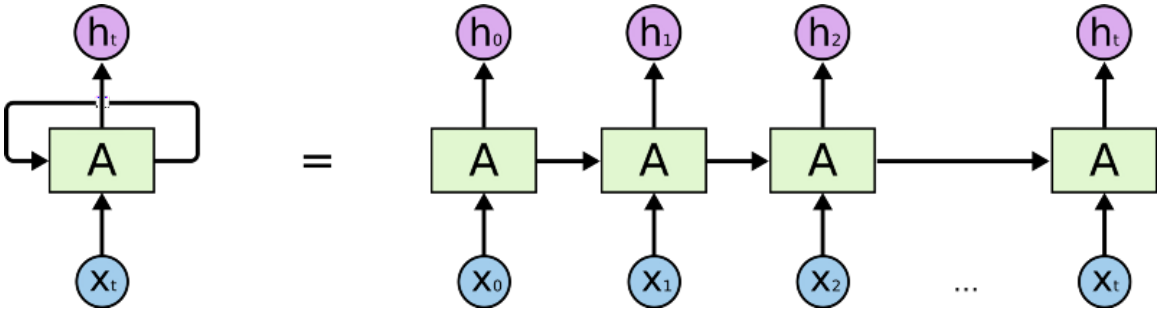


Figure 2.11: An unrolled recurrent neural network that allows persistence of information. Available at: `www.socs.binus.ac.id/2017/02/13/rnn-dan-gru/`. Accessed on 02 June, 2018.

But not everything is all roses, and despite being quite efficient, this type of neural networks has a significant problem. The RNNs can indeed use the past context to perform a present task, but just when the gap between the relevant information and the place that it is needed is not very long. Unfortunately, as that gap grows, RNNs become unable to learn and connect the information. This problem was already explored in depth by Yoshua Bengio et al.[71] and by Sepp Hochreiter et al.[72] and they found the problems that are causing this (but they are not relevant for this project).

Thankfully, after Hochreiter et al. studied this case, they introduced a special type of recurrent neural network without this problem - the so called LSTMs (long short term memory

[11]It is a difficulty found in training certain Artificial Neural Networks with gradient based methods, such as back-propagation.

networks). So, LSTMs were explicitly designed to avoid the long-term dependency problem. They have the nature to remember information for very long periods of time, not needing to struggle to learn them [72].

As said before, currently many projects use this type of neural networks to classify texts [73][74][75] and it is outperforming many other models.

2.3 Sentiment analysis research

There have been several research projects that apply sentiment analysis to different Web 2.0 sites in order to extract general public sentiments.

Apoorv Agarwal et al., for example, look at Twitter and build models to classify "tweets" into positive, negative and neutral sentiment. They build models for two classification tasks: a binary and a 3-way task. The first wants to classify sentiment into positive and negative classes, the second one objective is to classify sentences into positive, negative or neutral classes [26].

For all the experiments they use Support Vector Machines (SVM) trained using 5-fold cross-validation test results. The parameter C was tuned for SVM using embedded 5-fold cross-validation. For each classification task, they present three models (and combinations of them):

- Unigram[12] model;
- Tree Kernel model;
- 100 senti-Features model;
- Kernel and senti-Features;
- Unigram and senti-Features.

For the Binary classification task, the the best results were obtained with a SVM using the tree kernel and the unigram plus Senti-Features model (73.93 and 75.39 of acc).

For 3-way classification task, SVM using the tree kernel and the kernel plus Senti-Features model was the best model (60.60 and 60.83 of accuracy).

Alec Go, Richa Bhayani and Lei Huang for automatically classify the sentiment of Twitter messages (training data consists of Twitter messages with emoticons, which are used as noisy label) used a SVM, Naive Bayes and MaxEnt. The SVM (using bigram[13] and/or Unigram features) outperformed the other classifiers (82.2 of accuracy) [28].

Yujie Lu among others used English Twitter datasets to test different models (a SVM, a CNN, and a RNN) to see how well they perform in a real-world and experimental setting (for them, the experimental setting regards Twitter sentiment analysis just as a binary classification task without specified topics, in its turn the real-world setting regards it as a 3-class classification task with specified topics) [27].

For the binary and also 3-class classification the best results are obtained using SVM (unigram plus bigram) as model.

[12] Just 1 item from a given sample of text or speech.

[13] Sequence of 2 items from a given sample of text or speech.

Yoon Kim used CNN (built on top of *word2vec* [14]) on sentence-level sentiment classification and, despite little tuning of hyper-parameters, achieved very good results (with an CNN-non-static obtained an accuracy of 81.5). This CNN was used to classify movie reviews (positive/negative) with one sentence per review. In this task, other methods using SVM or Naive Bayes achieved worse results. But for example, for subjectivity dataset where the task is to classify a sentence as being subjective or objective the best results were obtained using a Naive Bayes, SVM and Multinomial Naive Bayes with uni-bigrams from Wang and Manning [30] [29].

In its turn, Xin Wang et al. outperformed several feature-engineering approaches by using LSTM recurrent network to predict polarities of tweets (as a bonus LSTM also captured the special function of words) [31].

The best ranking teams in subtask A of SemEval-2017 [15] Task 4: Sentiment Analysis in Twitter (Given a tweet, decide if it expresses positive, negative or neutral sentiment) for the English dataset used an ensemble of LSTMs and CNNs with multiple convolution operations or a LSTM with an attention mechanism (there are a lot of training data). Naive Bayes had the best results for Arabic dataset, where there are few data.

Every year there are teams that compete to achieve the best possible results in SemEval tasks. In the last year (2017) 48 teams participated in the task 4 of SemEval, 20 of them used deep learning methods (like CNN and LSTM networks). SVM were also very popular, several teams (for example, three of the top-10 (INGEOTEC, SiTAKA, and UCSC-NLP)) adopt this classifier combined with network methods or SVMs with dense word embeddings features. There were also teams using other classifiers such as Maximum Entropy, Logistic Regression, Random Forest, Naive Bayes, and Conditional Random Fields (CRFs). Tweester, for example, used Random Forest and stayed in sixth [32].

In SemEval-2016 Task 4, most of the teams who were ranked at the top also used deep learning, like CNNs, recurrent NNs, and word embeddings. In terms of supervised learning methods, the most used were deep learning (LSTMs, in particular). The methods based on kernel machines are becoming less and less used, and other methods are practically non-existent. For the same subtask A, the top-scoring team (SwissCheese) trained an ensemble of 2-layer convolutional neural networks (CNNs) whose predictions are combined using a random forest classifier. They use different types of filter shapes, pooling shapes and usage of hidden layers. Word embeddings generated via word2vec, and distant supervision were used as inputs to train the neural networks [33].

This type of supervision uses an existing database, such as Freebase or a domain-specific database, to collect examples for the relation that people want to extract. Then these examples are used to create training data automatically [34].

The team ranked 2nd (SENSEI-LIF) also used CNNs: they fed a final network using embeddings trained with different units (like sentiment and lexical). To create these embeddings, they used several auxiliary neural networks, one for each input space, and trained them separately to be able to extract representations from their hidden layers (polarity embeddings), after they used these polarity embeddings by concatenating and feeding them to a final CNN [35].

[14] Group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks trained to reconstruct linguistic contexts of words [36].

[15] SemEval (Semantic Evaluation) is characterized by having a continuous series of evaluations systems of computational semantic analysis, being organized under the domain of SIGLEX (Special Interest Group on the Lexicon of the Association for Computational Linguistics).

For subtask B (Given a tweet known to be about a given topic, the objective is to predict whether it conveys a positive or a negative sentiment towards the topic), the top-scoring team was Tweester. They used a combination of CNNs, semantic-affective and topic models, and word embeddings trained using *Word2Vec* [37].

For this task in particular, five of the top-10 participating teams used convolutional neural networks, and the others three used recurrent neural networks or SVMs. Seven of these teams incorporated in their systems word embeddings.

Besides these mentioned tasks, there are many others that are guided by the SemEval, for example the subtask 4 of SemEval-2014 Task 4 ("Aspect Based Sentiment Analysis") attracted also very teams. For this subtask, aspect categories (like food, service, etc.) are provided for each review sentence and the goal is to determine the polarity (positive, negative, conflict, or neutral) of each of them. The top team (NRC-Canada) used an SVM classifier trained with several features mainly based on n-grams, parse trees, and several dictionaries/lexicons [38]. The second best team also used a SVM (XRCE), trained by features related to information from its syntactic parser, BoW features, and an out-of-domain sentiment lexicons [39].

2.4 Data manipulation and Opinion Mining

2.4.1 Pre-processing of data

As was said ago, the datasets before being input to the classifier, have to be treated (pre-processed) to extract useful information (for example, to distinguish opinions from facts and then analyze only opinionated sentences it is a useful data pre-processing).

Apoorv Agarwal et al., for example, replaced all the emoticons with their sentiment polarity by looking up an emoticon dictionary, URLs with a tag "U", targets (e.g. "@John") with tag "T", negations by tag "NOT" and a sequence of repeated characters by three characters, for example, cooooooooool to coool. Next, they tokenized the tweets using the Stanford tokenizer [46] and excluded all the stop-words[16][26]. After this work on the original data they created different input features based on the processed data.

The NRC-Canada team that participated in the subtasks of SemEval-2014 task 4 also treated the data. For subtask 1, they tokenized the sentences to eliminate punctuation and then the token sequence was tagged using a semi-Markov tagger [17]. In subtask 2, sentences were first tokenized and parsed with the Stanford CoreNLP toolkits[18] to obtain part-of-speech (POS) tags and (collapsed) typed dependency parse trees. Sentences on subtask 3 and 4 were also tokenized, stemmed (with Porter stemmer) in the first and part-of-speech tagged, with CMU Twitter NLP tool, in the second one [38].

In its turn, DLIREC team tokenized and parsed (using the Stanford Parser[19]) all the sentences [49].

Bin Lu and their co-workers worked on *TripAdvisor* hotel review corpus. For each review, this corpus contains an associated overall rating, as well as ratings for 7 aspects (like value, room, cleanliness, among others). This aspect-rating was used to remove reviews missing any of the first 6 aspects, and they also excluded reviews that were too short or too long.

[16]The most common words in a language, usually (like of, is, on, at, etc.).

[17]The tagger had two possible tags: "O" for outside, and "T" for aspect term, where an aspect term could tag a phrase of up to 5 consecutive tokens

[18] http://nlp.stanford.edu/software/corenlp.shtml

[19]http://nlp.stanford.edu/software/lex-parser.shtml

After, datasets were tokenized and sentence splited using the Stanford POS Tagger. For topic models, they removed singleton words, and stop words [48].

Most of the showed teams used the Stanford NLP toolkits to tag and parse a sentence, but exists many more. *SyntaxNet*, for example, is a framework for a syntactic parser, which is a key first component in many natural language understanding (NLU) systems. Given a sentence as input, it tags each word with a part-of-speech (POS) tag that describes the word's syntactic function, and it determines the syntactic relationships between words in the sentence, represented in the dependency parse tree. To a better perception of the meaning of a parser we can attempt at Figure 2.12 [50].

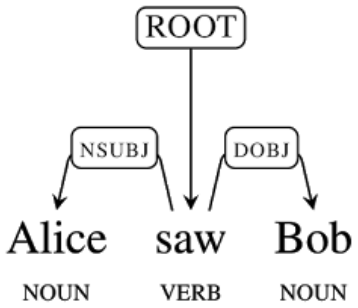


Figure 2.12: Simple dependency tree.
Available at `www.ai.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html`. Accessed on 30 January, 2018.

As shown above, this data pre-processing is fundamental to create features that will be used as input to the classifier, or to create other resources, like dictionaries. For example, Huan-Yuan Chen and Hsin-Hsi Chen created an aspect dictionary after tokenizing, parsing and tagging the sentences. The conditions to a word were regarded as an aspect term candidate were the following: its POS tags[20] is NN (Noun, singular or mass), it occurs at least 100 times, it is accompanied with an opinion word within the same segment, and (4) their dependency is nsubj (nominal subject) [45].

But what are dictionaries and what are they used for?

2.4.2 Dictionaries and other approaches

Dictionaries are a list of words, often arranged alphabetically, of one or more specific languages. The words of this list are usually associated with some useful value, like information (like positive or negative scores) or definitions, usage, etc., or can be a book of words in one language with their equivalents in another. They are sometimes also known as a lexicons[21].

So, a crucial list of words for any kind of text mining is a dictionary. Currently, and as was said before (in section 2), when people develop software applications for sentiment analysis or opinion mining there are basically two main options:

[20] Part-of-speech tagger processes a words sequence, and attaches a part of speech tag to each word.
[21] List of words that belong to a particular language.

- Assign sentiment values to text or text parts, making use of opinion dictionaries (general sentiment lexicons-based approach) [51];

- Inferring rules (and sentiment dictionaries), e.g. using machine learning techniques (like SVMs or neural networks), from previously annotated documents such as product reviews annotated with an overall product judgment (machine learning-based approaches) [51].

These two options can also be combined (hybrid approach). This approach is very useful because words may evoke different emotions in different contexts, and the emotion evoked by a phrase or a sentence is not simply the sum of emotions conveyed by the words in it, so it is necessary to use the first approach combined with any sophisticated emotion detecting algorithm (like machine learning methods) and this emotion lexicon will be a very useful component for the algorithm to detect the polarity of a sentence/aspect [59].

Let us firstly attempt on the first approach. To classify a sentence as positive or negative, for example, the opinion lexicon approach is very useful. These approach depends on finding the opinion lexicon used to analyze the text. There are two main methods for doing this: (1) The dictionary-based approach which depends on finding opinion seed words[22], and then searches into a dictionary (like *WordNet*[23] or *thesaurus*[24]) of their synonyms and antonyms. (2) The corpus-based approach begins with a seed list of opinion words, and then finds other opinion words in a large corpus, to help in finding opinion words with context specific orientations. This could be done by using statistical or semantic methods [52].

These approaches made by hand are very time consuming and because of that they are not used alone. One of the above two approaches is usually combined with an automated method as a final check to avoid the mistakes that resulted from the automated methods [52].

Nowadays there are already many online dictionary repositories (like *SentiWordNet*[25], *Vader*[26], and *Sentilex-PT*[27]) and it is not necessary to create a dictionary from the root. *SentiWordNet* is a public lexical resource and is the result of the automatic annotation of three sentiment scores: positivity, negativity, neutrality to all the synsets of *WordNet*. Each synset s is associated to their three numerical scores which indicate how positive, negative, and neutral the terms contained in the synset are [53]. *Vader* (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media, and works well on texts from other domains [55]. *SentiLex-PT* is a sentiment lexicon designed for the extraction of sentiment and opinion about human entities in Portuguese texts [54].

In addition to opinion dictionaries there are others that can be used in the field of sentiment analysis, such as aspect dictionaries. These type of dictionaries can be used when the objective is to detect what are the existing aspects in a sentence and determine the polarity of each aspect term or when we want to extract features based on them. They can be categorized or not, this means that the terms of a lexicon can be associated with their aspect category (for example, dessert belongs to food category).

[22]Set of opinion words that is collected manually with known orientations.

[23]It is a large lexical English database.

[24]It lists words that are grouped together according to similarity of meaning (like synonyms and antonyms), contrary to a dictionary.

[25]Available at http://sentiwordnet.isti.cnr.it/.

[26]Available at https://github.com/cjhutto/vaderSentiment.

[27]Available at http://xldb.fc.ul.pt/wiki/SentiLex-PT01.

Then, to be able to do useful work with the dictionaries or extract terms (like aspect or opinion terms) from a sentence and save them into a dictionary (or list) it is necessary to have a method that identifies the terms that are associated to opinions, aspects, categories, among others, or a method that match a term of a sentence with a term of a specific dictionary.

To find the terms that are associated to a dictionary term we can do a simple matching, but it is well known that this literal term matching has severe drawbacks, mainly due to the ambivalence of words and their unavoidable lack of precision as well as due to personal style and individual differences in word usage, among other things [56]. So, it is usually necessary to use semantic, lexical and/or statistical methods that allows us to find terms that have the same meaning (or are synonyms) as the dictionary terms. For instance, Maite Taboada and others stemmed the words tagged as an adverb and tried to match them to an adjective in their main dictionary [57].

Stefano Baccianella et al. to create the *SentiWordNet* 3.0 lexicon used the following 2 techniques:

- Synset term matching: If a Micro-WN(Op)[28] synset Si, that is not in their dictionary, contains exactly the same terms of a WORDNET 3.0 synset Sj , and such set of terms appears only in one synset in both WORDNET 2.0 and 3.0, they consider Si and Sj to represent the same concept;

- Gloss similarity: For each Micro-WN(Op) synset Si that has not been mapped, they compute the similarity between its gloss and the glosses of all WORDNET 3.0 synsets, where a gloss Gi is represented by the set of all character tri-grams contained in it. These similarity is computed via the Dice coefficient [53].

Hossam S. Ibrahim and their companions for detecting and extracting idioms and proverbs from text used N-gram and lexical similarity techniques. First, all possible phrase sequences are extracted from the topic as most Arabic idioms/proverbs not less than two words and measure the similarity between the extracted phrases and the idioms in the lexicon using cosine similarity[29]. The detected phrase is considered as candidate idioms when its cosine similarity value is more than (0.7) which mean that more than half of its terms are similar.

They also used another similarity measure method which is Levenshtein distance[30] to determine the correct idiom phrase from the candidate phrases (obtained by the cosine similarity) and to remove noise phrases (not idiom phrases, with cosine similarity more than or equal to 0.7) [65].

As said before, if there is no dictionary or just a small one, its necessary a method that extends/creates a dictionary by identifying the correct terms (for example aspect terms for a aspect lexicon) or a machine learning method that identifies the aspects that are intended to analyze or that expresses something (or a junction of these two).

Apoorv Agarwal et al. used an emoticon dictionary and an acronym dictionary. To create the emoticon dictionary they labeled 170 emoticons listed on Wikipedia with their emotional state (for example: ":)" labeled as positive. The acronym dictionary was compiled using

[28]Consists of 1,105 synsets manually annotated by a group of five human annotators.

[29]Cosine similarity measure the similarity between two texts, given the value of range (1,-1), value of (1) for identical texts and (-1) for opposite texts.

[30]Levenshtein distance is a string metric for measuring the difference between two sequences of text. It is the min number of single character edits required to change one word into another using three actions (insert, delete, substitution). So, the bigger the return value is, the less similar the two words are.

an online resource (https://www.noslang.com/). This dictionary have translations for 5,184 acronyms (for example, lol was translated to *laughing out loud*)[26].

Bin Lu and his collaborators used a sentiment lexicon introduced by T. Wilson et al.[47] to remove singleton words, and stop words that do not appear in these lexicon [48].

The objective of Huan-Yuan Chen and Hsin-Hsi Chen was to extract implicit opinions (no opinion words can be used as clues) from Chinese hotel reviews and identify their implicit aspects and polarity [45].

They extracted opinions from Chinese hotel reviews, after that transferred polarity and aspect from explicit expressions to the corresponding implicit opinions and use that opinions to train aspect and polarity classifiers (SVM and CNN).

This work bases on the fact that an implicit opinion and its neighbor explicit opinion tending to have the same aspect and polarity.

An opinion dictionary (opinion words associated with their polarity) and an aspect dictionary[31], based on the first one, were created. Words of POS tags, like predicative-adjective (VA) or Copulas (VC) are candidates to be an opinion word.

NRC-Canada team, from the Yelp restaurant reviews corpus, created an in-domain sentiment lexicon and a lexicon of terms associated with the aspect categories of food, price, service, ambiance, and anecdotes for restaurants. To create these two lexicons and extend them with more terms, they used the PMI[32] algorithm [38].

Mohammad and Turney created a high-quality, moderate-sized emotion lexicon by manual annotation through Amazon's Mechanical Turk service[33] (this lexicon was carefully chosen to include some of the most frequent nouns, verbs, adjectives, and adverbs). The annotations were manually done by *crowdsourcing*[34].

For these annotations, they first identified a list of wanted words and phrases, using for that the *Macquarie Thesaurus* as source pool for unigrams and bigrams (like noun unigrams, noun bigrams and verb unigrams) and from this list they chose those that occur frequently in the Google n-gram corpus [61]. They also ignored terms that occurred in more than one *Macquarie Thesaurus* category. Lastly, they chose all words from each of the six emotion categories in the *WordNet* Affect Lexicon that had at most two senses in the thesaurus [59][60].

Carlo Strapparava and Alessandro Valitutti created a *WordNet* Affect Lexicon has a few hundred words annotated with the emotions they evoke [35]. These lexicon was created by manually identifying the emotions of a few seed words and then by getting all their *WordNet* synonyms as having the same emotion [62][59].

Hu and Liu manually create a small list of seed adjectives tagged with positive or negative labels. Their seed adjective list is also domain independent. An effective technique is proposed to grow this list using *WordNet* [63].

DLIREC team implemented the Double Propagation (DP) algorithm[36] to identify possible aspect terms in a semi-supervised way. The terms identified were stored in a list for latter

[31]Not categorized.

[32]Pontwise mutual information - Given two random variables, A and B, their mutual information comes from the "amount of information" obtained on variable A, from variable B[58].

[33]An online *crowdsourcing* platform that is especially suited for tasks that can be done over the Internet through a computer or a mobile device.

[34]Model based on obtain services from a large, nearly open and often rapidly-evolving internet user groups to individuals or organizations.

[35]http://wndomains.fbk.eu/wnaffect.html

[36]It propagates information between opinion words and targets.

use. They also extended the opinion lexicon with addition seed opinion words by using the 75 restaurant seed words listed in Sauper and Barzilay [49].

For subtask 1 of SemEval-2014 task 4 (detect aspect terms in sentences), the NRC-Canada team used the in-house entity-recognition software, very similar to the system used by de Bruijn et al. to detect medical concepts. The best team (IHS RD.) relied on Conditional Random Fields (CRF) with features extracted using named entity recognition, POS tagging, parsing, and semantic analysis to extract these terms. They also used additional reviews from Amazon and Epinions (without annotated terms) to learn the sentiment orientation of words and they trained their CRF on the union of training data that were provided [38].

UNITOR team to detected an aspect category modeled this problem as a multi-label classification task where 5 categories (ambience, service, food, price, anecdotes/miscellaneous) must be recognized. In the constrained version, each class has been tackled using a binary multi-kernel SVM. A category is assigned if the SVM classifiers provides a positive prediction [64].

Chapter 3

Data collection and analysis

In this project, the data was obtained from two sources: *TripAdvisor* and *Kaggle*.

The data from the first source is obtained by extraction tools and is exclusively in Portuguese. The data retrieved from the second is a pre-defined *TripAdvisor* review dataset in English and will serve as a way of comparison to the results obtained by the classifiers with regard to the prediction of Portuguese comments rating.

3.1 Data extraction

In these last years, the computational treatment of opinion, sentiment, and subjectivity has attracted a great deal of attention, mainly due to its potential use for applications. But for this treatment to be reliable it is necessary to have a significant amount of data.

Social media and review sites in these days are the trends and people communication with these kinds of tools is increasing. The data that is produced by these platforms allows to evaluate objectively the people opinion and to identify strengths to be profitable and weaknesses to improve, and because of that, it is a good sentiment analysis subject.

In this project, the exploited data was extracted from Portuguese *TripAdvisor*, a review site. This happened because the Portuguese datasets presented in online repositories are really scarce, so, they needed to be obtained directly from the source.

To complete this extraction was initially tried the manual copy and paste approach, but the time spent was absurd (more than 10 minutes to get all the reviews from a single hotel). Automated extraction was then tested and in only a few seconds could get the job done.

As the web pages that needed to be analyzed and the information that needed to be extracted were known, instead of web crawling, the web scraping (also called Web Data extraction, Screen Scraping, Web Harvesting) was the tool used to do the automated data extraction.

There are many web scrapers, but for this project the chosen on is an open source and collaborative framework that is given by the name of *Scrapy*. This choice lied in its compatibility with *Python* and in its fully-fledged solution that allows writing small amounts of *Python* code to create a *spider* (an automated bot that can visit web pages and scrape contents of them).

3.1.1 Methodology to extract data

Firstly, the pages of the Portuguese *TripAdvisor* domain were scraped to obtain all the hotel's and restaurant's links from several places (as can be seen at Tables 3.2 and 3.3). After obtaining these links, they were also scraped to collect all their reviews and other important pieces of information (like user's ratings).

The hotel/restaurant links and reviews from different regions were initially separated at different *.txt* and *.csv* files, respectively, to make it easier to create the tables that have information about their number of hotel and reviews. Later, they were all joined into a file of their respective type.

These datasets will be used for classification (rate a comment as positive or negative by assigning it a score between one and five) and opinion mining (associate hotel parameters (e.g. room, bed, restaurant) to feeling markers (e.g. great, fantastic, bad) and give them a punctuation).

Apart from the Portuguese data and to compare the results obtained with this data classification was used a dataset of London-based restaurants' *TripAdvisor* reviews[1], obtained from website *Kaggle*.

3.1.2 Amount and characteristics of Portuguese data

As is evident from Tables 3.3 and 3.2, the number of different regions for hotels is much higher then for restaurants, this is due to the number of reviews per restaurant being extra superior to the number of reviews per hotel, so there was no need to extract reviews from more restaurant regions.

For hotels, the regions were chosen to cover most of the Portuguese territory and a few regions of different continents (like Rio de Janeiro, Londres and Camboja). For restaurants, the objective was only to get reviews and other important information of Portugal capital (Lisbon) and Aveiro district.

The data scraped from each hotel/restaurant and saved at a *.csv* file was:

- hotel or restaurant name, rating and locality.
- review title, rating, text, date and link.

Let us now turn our attention to the reviews. All of the information related to the reviews obtained from *Scrapy* are originally Portuguese - this means that all of the comments automatically translated to this language were discarded.

For example, the *BessaHotel Liberdade* situated in Lisbon supposedly has 157 comments in Portuguese but 17 of them are automatic translations[2].

Finally, as an important note, Table 3.1 shows the class distribution of all Portuguese comments obtained.

[1] Number of English *TripAdvisor* reviews obtained from *Kaggle*: 20 000.

[2] This information was obtained from $https://www.tripadvisor.pt/Hotel_Review-g189158-d7690352-Reviews-or135-BessaHotel\_Liberdade-Lisbon\_Lisbon\_District\_Central\_Portugal.html$ and accessed on 17 April 2018.

Review's rating	Frequency
1	15 266
2	18 109
3	41 469
4	120 819
5	146 471
Total	342 134

Table 3.1: Distribution of comments ratings for Portuguese dataset from one to five.

3.1.3 Number of reviews, hotels and restaurants obtained

	Nº of restaurants	Nº of reviews
Aveiro District	1 053	38 325
Lisboa District	3 842	187 399
Total	4 895	225 724

Table 3.2: Number of restaurants and user reviews retrieved from *TripAdvisor* on 13 April 2018.

	Nº of hotels	Nº of comments
Aveiro District	287	5 820
Coimbra District	293	8 779
Lisboa District	1 612	47 980
Porto District	839	21 422
Algarve region	1 715	27 409
Covilhã city	5	1 421
London	5	227
Camboja	5	104
Others	17	3 248
Total	4 778	116 410

Table 3.3: Number of hotels and user commentaries retrieved from *TripAdvisor* on 13 April 2018.

3.2 Amount and characteristics of English data

The dataset collected from *Kaggle* comprises exactly 20 thousand comments (associated with their respective title, rating, hotel, among others).

As previously mentioned and to recall, these data will be used to train classification models, serving as a comparison to the Portuguese data that is also used for this purpose.

The distribution of comment ratings can be seen in Table 3.4. Please note that comments with no associated score were discarded.

Review's rating	Frequency
1	1 027
2	1 144
3	2 122
4	5 287
5	7 645
Total	17 225

Table 3.4: Distribution of comment ratings for English dataset (1 to 5).

3.3 Dataset of associations

In addition to the datasets mentioned, there is another one. This dataset contains 722 examples, all of them are constituted by an association between an aspect's and opinion's word, by its classification (from one to five) and the commentary that it belongs, among other useful information.

The creation of these examples was done through the algorithm explained in chapter 4, section 2.

After obtaining and saving the examples into a *.csv* file, they were manually analyzed to saw their veracity and to correct possible errors.

This data was used to train, validate and test a classifier that predicts the sentiment of an association expression (for example, to the 'comida muito boa' expression was attributed the value 4).

The final distribution by classes can be seen in table 3.5.

Association's class	Frequency
1	30
2	91
3	205
4	221
5	176
Total	723

Table 3.5: Distribution of association's classes (1 to 5).

3.4 Pre-processing of data

The data obtained from the web scraping, before being used by machine learning classifiers, opinion mining, sentiment analysis, etc. needs to suffer some kinds of pre-processing with a focus on reach positive outcomes. As is often said no quality data, no quality results - garbage in, garbage out.

Quality decisions must be based on the quality of the data! e.g., duplicate or missing data may cause incorrect results!

Data preparation, cleaning, and transformation comprises a lot of the work done in a data mining or sentiment analysis application.

3.4.1 Pre-processing data to classifiers

For this project, the data in front of being inputted into classifiers need to face some mutations, like usually needs.

For this task, was used two types of datasets. One obtained from *Kaggle*, which contains about 20 thousand comments in English (as said before), and the other one obtained from web extraction, with all of the reviews in Portuguese.

So, before a classifier receives the training data and predicts a rating (1 to 5) given by a person in reliance on their review, all of the review letters are ratified to lowercase.

For English dataset, all the characters that were not digit, letter, point, comma and plica were expelled.

In turn, the Portuguese language is more complex and special care must be taken when we are replacing characters. For this language, all the characters were maintained, with the exception of some graphics signals (like asterisk and median point), all the mathematics, monetary and other typographic symbols.

Following this came the tokenization, all of the review words were separated (using the space character as the splitter) and aggregated into a *Python* array[3]. Then this array was turned into a numerical one: whose number of this new array represents a word of the last. An example of this can be seen in image 3.1.

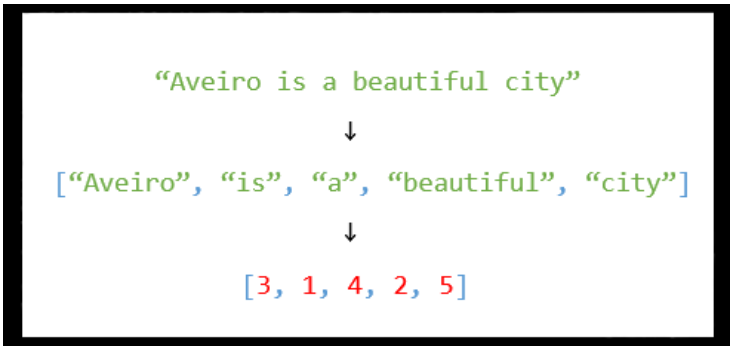


Figure 3.1: Example of tokenization done in data pre-processing.

After this, we were stuck upon the fact that the extensiveness of the array numbers was different amongst distinct reviews. So, a method to put them into the same length was needed. This was done by padding each one of the number chains - zeros were placed at the top of the array until it was the same length of the longest comment array.

The model will learn that these zero values carry no information so indeed the sequences are not the same length in terms of content, but same length vectors are required to perform the computation.

[3]Data structure that contains a group of elements.

For closing this stage was only required to explore the training ratings that were associated with the user reviews - which are the output of the classifiers (label classes).

For multi-class classification, the Portuguese dataset was alright and didn't need to suffer modifications. In its turn, the English one had their ratings in a string format - e.g. 4 of 5 bubbles. So, they needed to be born-again as numbers. For this to happen, just the first character of the ratings (4 in the last example) was taken into account and converted by *Python* to a number format.

For binary classification, the reviews of both datasets classified below three (including) were classified as zero (bad reviews) and the others (greater than three) as one (good reviews).

3.4.2 Pre-processing data to find associations

The data collected from *TripAdvisor* through *Scrapy* had some gaffe. For example, *Scrapy* sometimes adds more than one space between words and/or eliminates the space within a word and a dot or comma. If these errors were not corrected the *SyntaxNet* produced erroneous results.

After this correction to be fathered, the *SyntaxNet* was then applied to the reviews and were obtained annotations to their every word. An illustration of these word's annotations can be noticed at Figure 3.2. This tool tokenizes a sentence, tags it and gives associations between words included in that sentence.

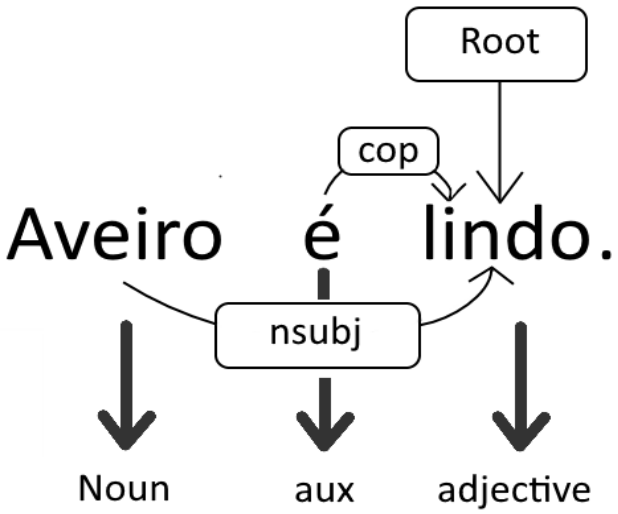


Figure 3.2: Example of annotations retrieved by *SyntaxNet* for the text *'Aveiro é lindo'*

3.4.3 Dictionaries

The tasks related to opinion mining will need the existence of two types of dictionaries:

- An opinion dictionary that contains adjectives like *lindo*[4] and *bom*[5].
- An aspect dictionary that includes aspects related to a hotel or a restaurant, such as *comida*[6] and *serviço*[7].

The doubt here was to know if these types of dictionaries for Portuguese already existed or needed to be created from scratch. For the opinion lexicon, was found and chosen an interesting option - *SentiLex-PT*.

SentiLex-PT is a lexicon of Portuguese feelings consisting of 6,321 adjectival lemmas (by convention, in the singular masculine form) and 25,406 flexed forms (each entry is represented by a form inflected in gender and number, which is associated with the respective lemma as we can see at Figure 3.3)[66].

Their entries correspond to human predicative adjectives that were compiled from various public resources. Below we can see the attributes of each entry (file line) in the lexicon:

- The polarity of the adjective.
- The target of feeling.
- The method of assigning polarity.

```
abalada,abalado.PoS=Adj;FLEX=fs;TG=HUM:N0;POL:N0=-1;ANOT=JALC
abaladas,abalado.PoS=Adj;FLEX=fp;TG=HUM:N0;POL:N0=-1;ANOT=JALC
abalado,abalado.PoS=Adj;FLEX=ms;TG=HUM:N0;POL:N0=-1;ANOT=JALC
abalados,abalado.PoS=Adj;FLEX=mp;TG=HUM:N0;POL:N0=-1;ANOT=JALC
```

Figure 3.3: Four entries of the *SentiLex-flex-PT02.txt*, referring to the Portuguese shaken lemma.

For this project only was needed the inflected form of the lemmas and its polarity (-1 for negative words and 1 for positive). The file *Sentilex-flex* also has Portuguese expressions of opinion, such as the example present into Figure 3.4 and initially, these opinion expressions were also used but a quick search for the mentioned expressions into the reviews revealed that there were very few matches. As a consequence, the expressions were laid away and only remained the inflected forms.

```
andou com cara de caso,andar com cara de
caso.PoS=IDIOM;Flex=J4s|P3s;TG=HUM:N0;POL:N0=-1;ANOT=MAN
```

Figure 3.4: One entry of the *SentiLex-flex-PT02.txt*, referring to a Portuguese expression.

After the expressions discard, some of the opinion words that were not relevant for this project were also dropped. *Apropria*[8] and *calar*[9] are some the words that were throw out. But to pay up, feeling words considered relevant in the area of hotels and restaurants were added to improve the contents of the dictionary (like the word *grande*[10]).

[4]Portuguese adjective. In English these means beautiful.
[5]Portuguese adjective. It means good in English.
[6]Portuguese word. It means food in English.
[7]Portuguese word. Translated to English it means service.
[8]It means appropriates appropriates.
[9]It means shut up.
[10]Means big in English.

Posteriorly, the objective was to find two aspect dictionaries, the two were a mix of words, one of them with words related to hotels and the other one related to restaurants. But after a tolerable research, no dictionary of this kind was found.

To build these dictionaries some regular expressions[11] were applied into the annotations outputted by *SyntaxNet*. But before these regular expressions hit town, the *TensorFlow* toolkit was applied toward 16 000 hotel and restaurant reviews (equally divided) and for each review, the annotations created had information about all their words. Just then these explanatory notes were hit by some *regex* to catch on possible aspects words. These good prospects were then automatically inserted into two distinct dictionaries files (*.txt* extension) - hotel and restaurant aspects - with their number of appearances through the reviews.

The regular expressions applied were:

- <NOUN>*<NOUN>
- <NOUN> (<ADJ>|<NOUN>)*
- ((<ADJ>|<NOUN>)+|((<ADJ>|<NOUN>)* <ADP>?)(<ADJ>|<NOUN>)*) (<ADJ>|<NOUN>)

Where the Universal POS tag NOUN represents a noun, ADJ an adjective and ADP an adposition. The symbol * represents zero or more times, + represents one or more times, and | represents alternatives.

In other words the second regex, for example, means NOUN followed by a sequence (0 or more times) of ADJ or NOUN.

After the dictionaries creation and saving, these outputted files were analyzed to see if there were false positives and after a quick glance, the following sentence was confirmed: Nothing is all good. And nothing is all bad. The generated files had very good matching aspects but had also some bad results.

These misrepresentations were then mostly automatically eliminated, by using a conceived *Python* script for this purpose. The posterior persisting mistakes were eliminated by an exhaustive analysis through all the dictionary words.

At the same time as the errors were eliminated, the aspects were also grouped into their general aspect. For example bread was included into food general aspect and reception into service. We can see the form of the dictionaries (that are *.txt* files) through Figure 3.5.

[11] A concise and flexible notation for finding and replacing patterns of text.

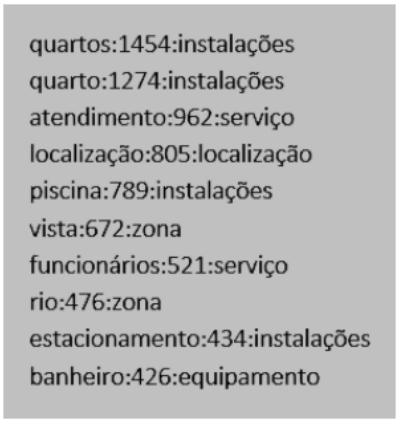


Figure 3.5: Ten entries of *hotelAspects.txt*, referring to Portuguese hotel aspects. The format is ***<aspect:appearances:generalAspect>***.

In order to conclude this stage, it is important to raise the dictionaries final size. The opinion dictionary got 47 318 sentiment words, the hotel and restaurant aspect glossaries harvested 1 506 and 1 191 terms, respectively.

Chapter 4

Sentiment Analysis

After the data has been obtained and pre-processed, the useful and touchable work begins now. The final data will be used by Machine Learning methods to classify comments and frameworks that allow the achievement of associations between specific words. Then these results will be displayed in a web interface.

As expected, the next sections will describe the work done to complete the tasks mentioned above.

4.1 Classification

There are no secrets and no magic in machine learning. In fact, there is an identifiable shape of practical and simple techniques that produce useful information from an ocean of data.

In this section, the structures that were constructed for some machine learning methods will be described, and with this, you will see that there is no magic, for sure.

4.1.1 Setup

So, in order to understand the body of a ML[1] method, it implies initially to know what is its desired output (the form of classification).

The first three machine learning classifiers used in this section outputted predictions associated with a rating (1 to 5) given by a person in reliance on their review. This type of classification is named multi-class.

The last one (neural network method), was also used for binary classification - classify a review as positive or negative. Here, this classifier was trained to predict instances into one of the two classes (zero or one), contrasting with the previous task where the classifiers predict into one of multiple classes. This model was the chosen one because it got better results in multi-class classification, and was found as interesting and useful to also train and test it for binary classification.

All of the methods were implemented with the programming language *Python* and with two efficient tools, *scikit-learn*[2] and *Keras*[3], for some of them were also used other *Python*

[1] Machine Learning.

[2] `http://scikit-learn.org/stable/index.html`

[3] `ttps://keras.io/`

libraries (that will be described further ahead).

Keras provides a high-level interface to the lower-level neural network library *Tensorflow*, greatly easing development [80]. In its turn, *Scikit-learn* provides simple and efficient tools for data mining and data analysis [81].

Developing these classifiers requires adopting a validation strategy that evaluates how well a model generalizes to independent examples. A popular approach is to split the data into training, test and validation subsets:

- The training subset contains the examples that the classifier uses to tune its parameters (like *alpha* to SVMs or the number of maximum trees for Random Forests);
- The validation subset is used to measure the training effectiveness and support the regularization techniques, such as early stopping;
- Finally, the test set is used to provide an impartial evaluation of the trained and validated model.

The data was divided into two (training and testing) subsets by using a *scikit-learn* method named *train_test_split*. This method split arrays or matrices into random train and test subsets.

To obtain the validation subset, the training one was again divided into two. To do this, the first three models used a grid search (provided by *scikit-learn*).

Grid searching works by running k-fold cross-validation with all combinations of the selected model's parameters and outputting the parameters which produced the best result (basically it works by tuning the hyper-parameters[4] of an estimator).

In its turn, the k-fold cross-validation operates by splitting the training data into k partitions, training the model with k-1 partitions and, finally, running the model on the validation set.

In the last model, it sets apart a fraction of the training data, does not train on it, and evaluates the loss and any model metrics on this data at the end of each epoch. This is done without needing the use of cross-validation, because *Keras* library gives an automatic way to do this.

4.1.2 Initial models

Now that what needs to be accomplished was known, it was necessary to get the idea of what were the best machine learning methods that could do that.

After a little research (that is described in Chapter 2, Section 2.2) was decided to apply the following methods: Naive Bayes, Random Forest, Support Vector Machine and Neural Network.

Of these four models only the two that had the best upshots continued to the next step, which was the improvement of the structure to obtain even better results. The dataset used to evaluate them in this phase was retrieved from Portuguese dataset (about 30 thousand comments).

Regard that this section only serves as the choice of the best models, the classifiers outcomes will not be shown in the results chapter.

[4]Parameters that are not directly learn within estimators.

Naive Bayes

The first method trained and tested was the Naive Bayes. From *scikit learn* library, some Naive Bayes algorithms were tested to see which was the supreme. These different algorithms differ mainly by the assumptions they make regarding the distribution of $P(x_1, ..., x_n|y)$.

For this specific classification task, the one that got the best results was the *MultinomialNB*[5].

In a simplified way, and just for settle facts, the difference between the traditional and the multinomial method is:

Whereas the simple one models a document with the presence and absence of particular words, the other one explicitly models the word counts and adjusts the underlying calculations to deal with them.

The final greatest model was achieved through a grid search, used to get the best results for the best parameters.

The search parameters were:

- *alpha*[6] from zero to one;
- *fit_prior*[7] that alternates between true and false.

To better understand the parameter *alpha* let us pay attention to how this algorithm works. In this model, the data is typically represented as word vector counts: $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$, where y represents a class (in this case from 1 to 5) and n is the number of features (in text classification is the size of the vocabulary) and θ_{yi} is the probability of feature i appearing in a sample belonging to class y.

The entries of the vector referred above are calculated by the following equation:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{4.1}$$

where N_{yi} is the number of times that feature i appears on class y (in the respective training set) and N_y is the total count of features appearing on the same class.

The parameter α (*alpha*) is a smoothing parameter. If the value assigned to it is greater than 0 then no feature will have zero probability in further computations. Setting $\alpha = 1$ is called *Laplace* smoothing, while $\alpha < 1$ is called *Lidstone* smoothing.

For both datasets, the k fold cross validation object used was a Stratified ShuffleSplit cross-validator. It is a merge of StratifiedKFold[8] and ShuffleSplit that returns stratified randomized folds.

To avoid misunderstandings, a question-answer pair should be given.

What is the difference between the Stratified ShuffleSplit cross-validation and a simple cross-validation?

The process of rearranging the data to ensure that each fold is a good representative of the whole is denominated stratification. For example in a binary classification problem where each class comprises 50% of the data, this method guarantees that in every fold, each class comprises around half the instances.

[5]Naive Bayes classifier for multinomial models.
[6]Smoothing parameter (0 = *nosmoothing*).
[7]Determines whether the previous probabilities of the class are learned or not.
[8]It is a variation of k-fold that returns stratified folds.

A simple cross-validation will only divide your data set into a pre-specified number of folds, and every sample must be in one and only one fold.

In its turn, shuffle splits will randomly sample the entire dataset through each iteration to generate a training and test set.

Finally, it is important to say that the number of splitting iterations was 5, and the validate data size for each split was 5 percent of the total.

Random Forest

The Random Forest classifier was the one that followed Naive Bayes method. It is represented in the *scikit learn* library as *RandomForestClassifier*.

This implementation merges classification trees by averaging their probabilistic prediction, instead of letting each one vote for a single class. Each tree in the whole is made from a sample drawn taking into account the training set (replacing it).

For this classifier was also used a grid search with the Stratified ShuffleSplit cross-validator to get its best parameters and results.

The parameters inputted in this grid were:

- Number of estimators (from 10 to 510);
- Maximum depth of the tree (from 2 to 14).

The number of estimators is the number of trees to be built before taking the maximum voting or averages of performance. Usually, a higher amount of trees results in a better performance but also makes the program slower.

The maximum depth of tree (number of nodes) is the maximum allowed depth of each tree in the forest, not the actual depth.

The default values for the parameters that control the size of the trees can lead to fully grown and unpruned trees that will potentially be very large on some data sets.

So, it is very important to control the hyper-parameters to get better results and to reduce memory consumption. The complexity of the forest and the size of the trees are examples of that.

To conclude, the number of splitting iterations was 5, and the validate data size for each split was 5 percent of the total (just as it was for its ancestor).

Support Vector Machine

The third model tried was the Support Vector Machines (SVM). This type of classifier tries to find the best hyperplane to separate the different classes, and this is achieved when the distance between the examples points and the hyperplane is at maximum, as referenced more closely in the state of the art chapter.

To complete the mission was used the *SVC* method. It is a function from *skicit-learn* capable of performing multi-class classification on a dataset.

To maintain the consistency a grid search needed to be used, and that's what happened. The number of splitting iterations was also 5, and the validate data size for each split was 5 percent of the total examples. Let's then see the parameters that were set to be tuned:

- Type of kernel (*rbf*, *linear* and *polynomial*)

- *Gamma* from 2^{-4} to 2^{4}
- C from 2^{-5} to 2^{5}

As we can notice by the explanation made in the Chapter 2, Section 2.2, the choice of what kernel to use is not easy, so it is essential to employ a grid search to tune this parameter.

The other two parameters are also very important to avoid overfitting and get better results. So, they needed to be tuned to make sure that the best model was actually obtained, and that was exactly what was done in this project.

Neural network

In this model, all of the comments (whose words were already being represented by numbers, see Chapter 3, Section 3.4.1, for more information) were mapped into a real vector domain, a popular and useful technique when working with text called word embedding. Unanue et al., Zhenzhen Li et al. and Tomas Mikolov et al. showed that pre-trained word embedding can improve the performance of systems and capture meaningful syntactic and semantic regularities [68][69][70].

The indices that represented the words in this technique were encoded as real-valued vectors in a high dimensional space. The similarity between the meaning of the words is translated to closeness in the vector space.

To this stage, each word was represented by a 50 length real valued vector and the total number of words that were considered for modeling equals to the 5000 most frequent words (the others had value zero).

Multi-class classification task. For the multi-class classification problem the labels were also changed. When we are modeling this type of classification using neural networks it is a good practice to reshape the label vector that contains values for each class value to a matrix whose elements are arrays with the size equal to the number of existing classes plus one where only the position corresponding to the class is one and the others are zero (for example, the output of a review with rating four is represented by [0 0 0 0 1 0]. This process is called one hot encoding or the creation of dummy variables from a categorical variable.

All of this was done using the *keras*, which provides a convenient way to convert positive integer representations of words into a word embedding by an Embedding layer and a way to transform the label vector into a categorical matrix. This layer was the first layer of the neural network model, as can be seen in Figure 4.1.

After the construction of this layer was necessary to choose the next one, and was decided to use an LSTM. Why that?

Taking into account the information learned in the state of the art research and past projects was also used LSTMs in this thesis. Therefore, following the embedding layer emerged then an LSTM with their memory units (smart neurons) matching up the size of the embedding layer output.

Finally, was used a Dense output layer with six neurons (zero to five classes) and a *softmax* activation function. *Softmax* squashes the layer's inputs into a probability distribution (divides each output such that the total sum of the outputs is equal to one). The output of the *softmax* function is equivalent to a categorical probability distribution, it tells the probability of any of the classes being true, for example if the predict for a review is class four, then the output could be something like [0.1, 0.1, 0.1, 0.6, 0.1].

So, the output of the *softmax* function is equivalent to a categorical probability distribution, and this makes it the more appropriated function to lead with this type of classification.

Then, because it is a multi-class classification problem and the labels are categorical, the categorical cross entropy was used as the loss function. This loss function entropy calculates the cross-entropy between an approximating distribution and a true distribution. Mathematically, this function computes

$$H(p, q) = -\sum_{x} p(x) log(q(x)) \tag{4.2}$$

where p is the true distribution and q the given probability distribution and x the random event.

Binary classification task. Contrarily to multi-class classification task, the labels stayed the same in binary task, zero for bad reviews and one for good ones. To this task the real examples labels (from one to five) had to be set to binary, and this was done by considering all the values below and equal to three as zero and the above as one.

For the output Dense layer, was used the *sigmoid* activation function (represented in Equation 4.3) instead of the *softmax* function. The *sigmoid* function takes any range real number and returns the output value which falls in the range of zero to one so that each neuron outputs a *Bernoulli* probability distribution independent of all other neurons.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.3}$$

Finally, the last thing different from the last classification task was the loss function: binary cross-entropy, instead of the categorical one, as is usually the case in binary classification scenarios. This function is represented by the equation:

$$L(y, p) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} y_{ij} \log(p_{ij}) \tag{4.4}$$

where i represents the review's index and j its label, p is the prediction outcome and y is the correct one.

Both tasks. The efficient *Adam* optimizer [76] was used for both tasks (multi-class and binary) and the training examples were inputted into the classifier in batches of size 32. The classifier used a regularization technique named early stopping with a patience of two epochs - 2 epochs without improving the loss score - and no minimum delta (how much need to improve from one epoch to another). In each epoch, one percent of the training examples were used to validate the loss's and accuracy's score. The total number of epochs for training had a value of 50 but in all the cases, because of the early stopping, the model stopped the training earlier (in average at the tenth epoch).

Choosing the Adam optimizer, instead of the more traditional Stochastic Gradient Descent, allows the adaptive tuning of the classifiers hyperparameters during the training and validation [77].

After getting the results (test accuracy, precision, among others), these were below the expectations and because of that it was needed to improve the model. So, to enhance the model's ability to learn higher-level temporal and sequential representations, another LSTM

layer was added. Also, this type of recurrent neural networks generally has the problem of overfitting, as a way to reduce it, Dropout layers were used between the two LSTMs and the last LSTM and the output layer.

Let's see how this layer works. Basically, during training, half of the neurons in a given layer are deactivated, but with the use of the dropout technique this generalization is improved, because it forces the layers to learn with different neurons the same "concept".

After training, during the prediction phase the dropout is shut off.

With these new additions it was also necessary to change the output of the primary LSTM. So, now it returns its full output sequences, although the second one only delivers the last step in its output sequence, leading to the loss of the temporal dimension (that is the same as saying that the input sequence was converted into a single vector).

More LSTM's and dropout layers were also tested in the model but the growth registered only occurred in the time, having results stay the same (results not shown). Because of that, these new additions were then removed, resulting in the model shown in Figure 4.1.

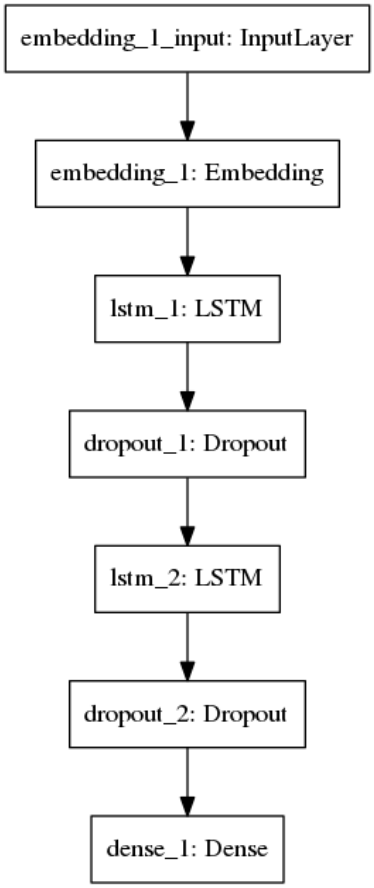


Figure 4.1: Neural network model to classify English and Portuguese reviews.

4.1.3 Final models

Of the four types of previous classifiers were only chosen the two best ones. As SVM and neural networks had better results than the other methods, they were the chosen ones and it

was then necessary to improve them.

Support vector machine

The first model to undergo for these improvements was the SVM. To get better results (accuracy, precision, among others) the parameters *gamma* and C were tuned using a higher range value. The values tested for these parameters were different for the two datasets (English and Portuguese). The English one got the following:

- Four values of C from 2^{-1000} to 2^{8} for *polynomial* and *rbf* kernels and from 2^{-200} to 2^{9} to linear kernels.
- Four values of *gamma* from 2^{-1000} to 2^{2} for *polynomial* and *rbf* kernels and from 2^{-200} to 2^{2} to linear kernels

Contrastively, for the Portuguese dataset, where the examples were much larger in number, the maximum range had to decrease. So, the range of values used was:

- Four values of C from 2^{-1000} to 2^{0} for *polynomial* and *rbf* kernels and from 2^{-200} to 2^{-2} to linear kernels.
- Four values of *gamma* from 2^{-1000} to 2^{-1} for *polynomial* and *rbf* kernels and from 2^{-200} to 2^{-4} to linear kernels

It should be noted that higher values for both datasets were tested than those in the above range, but these led to an excessive increase of the training time, and a low improvement of results.

The training inputs were also normalized using a *scikit-learn* method. Normalization is the process of scaling individual samples to have unit norm [82]. The unbalanced data was also corrected by giving to the smaller classes more importance than to the larger ones, doing this in an implicit way[9].

Neural network

As said before, most of the text classifications models opt to use word embeddings to represent tokens.

But contrarily to the initial neural network model, where the embedding used as supplied by *Keras* library, now the corpus of the embedding layer was created using *Word2vec's* skip-gram neural network model. To this model the number of terms considered to create the words embedding were all the different ones that appeared in the comments, and the vector representing them increased in size from 50 to 100.

The concrete implementation of *word2vec* used was the one from the *gensim* library[10].

In addition, the first dropout layer also changed their location in the model, being now placed between the embedding's and LSTM's layers. The final model is noticeable at image 4.2.

[9] This is done by defining a parameter of the grid search.

[10] https://radimrehurek.com/gensim/models/word2vec.html

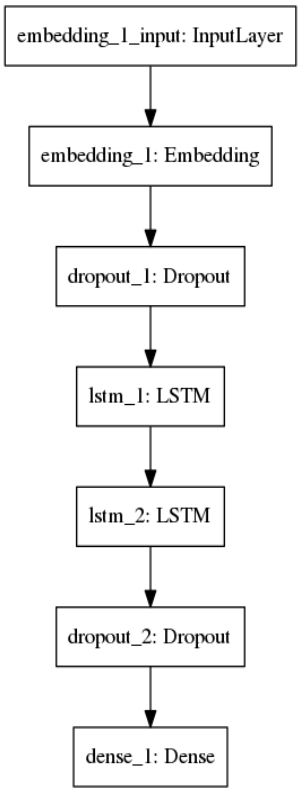


Figure 4.2: Neural network model to classify English and Portuguese reviews.

4.2 Opinion mining

Textual information around the world can be broadly classified toward two main types: truths and opinions. Truths are objective definitions of things and their characteristics. In contrast, opinions are ordinarily subjective expressions that describe people's sentiments, judgments, or impressions about entities, events, and their features.

The objective of this section was to extract relations between attractions aspects (truths) and sentiment words (opinions). Then this information was recorded into an indexing framework.

All of this section workflow will be done with the *Python* programming language.

4.2.1 Generate Graphs

To anchor this phase in a good harbor, the first step to be done was to take *SyntaxNet's* result and make it a graph easier to use.

Before presenting the *Python* library applied to create the graph, it is necessary to understand the output of *SyntaxNet*.

The *SyntaxNet* library used on *Python* gives for each comment an array whose elements are *Python* dictionaries. Each one of these dictionaries contains a word from a review, where this word starts and ends, their tag (for example 'noun'), its head[11] and the label associated.

[11] Associated word assigned by the *SyntaxNet* algorithm.

To create the graph, the output required for each element of a review's array was its head and term. Then in this graph was built an edge between one word and its head's word (being this done recursively until it reached the root word). Finally, the graph was successfully created when this was done for all the words in the comment.

NetworkX is the python library used to create these graphs. This library is capable of creating, manipulating and studying the formation, dynamics, and functions of complex networks [67].

It has many features, but the one that mattered to this project was its ability to generate classic graphs, random graphs, and synthetic networks. In this case, the graph created allowed multiple edges between any pair of nodes (the difference between this graph and a simple one can be seen in Figure 4.3). Finally, it should be noted that each connection has an associated weight (direct associations equals to weight one, for example).

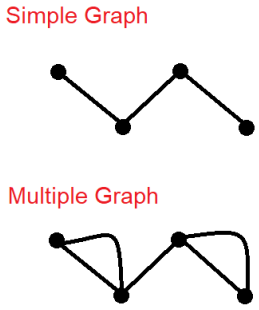


Figure 4.3: Difference between a simple and multiple graph.

This graph was very useful and efficient to identify the associations between aspect and opinion words. But to get these associations this kind of words needed to be identified first.

4.2.2 Identify the aspect and opinion words

This is the most time-consuming step[12] present in this section.

Let's start with the basics. To catch all the aspect and opinion review words it is necessary to extract from the three previously created dictionaries all their expressions.

After, the fastest way to search for these terms in the comment was needed to be chosen.

To accomplished that, the *re* (python module) was initially tried. This module provides many regular expression matching operations, but the one used in this project is given by the name *search*, which parameters are a word (in this case aspect or opinion) and a sentence (the review). This function output is of Boolean type - if matches true, else false.

The aspects were the first picked up and, as the dictionary was relatively small, this process was quite fast. The problem was when it came time to go through the dictionary of opinions. Because of its considerable size, the time it took to saw if each of its expressions existed in the text was relatively high - approximately 2 seconds to find all the review's opinions.

So much time was unthinkable, and because of that was opted to look for another module that would do the same but more quickly: after a little research *FlashText*[13] was then found.

[12]Speaking in computational terms.
[13]https://github.com/vi3k6i5/flashtext

This module can be used to replace or extract keywords in/from sentences.

After testing this module in some comments, was verified that a search which took about two seconds per comment with *re* module was reduced to 0.002 seconds.

FlashText was chosen and one more step completed. The aspect and opinions words that appeared in each comment were stored in 2 different *Python* dictionaries, the others were discarded.

After this, came the search and discovery of associations.

4.2.3 Obtain associations between aspect and opinion words

To complete this stage, the first thing done was going through the two *Python* dictionaries and for all combinations between the words of each one was checked the distance in the review between them. If there were more than 12 words in the array of review's tokens (created from *SyntaxNet* tokenization of the review) between one aspect and opinion, they were considered as not being associated.

This distance needed to be checked due to the incorrect writing of many comments, and because of this the result coming from *SyntaxNet* indicates that distant words in a comment are associated, and most of the time this is wrong (this fact was proven by observing 50 comments and all of their associations captured by the *SyntaxNet* dependency graph). The Table 4.1 shows us some examples of that.

Wrong associations	Comments
estacionamento muito bom	Excelente localização, um ótimo atendimento Café da manhã muito bom O apto triplo bem estruturado, dando privacidade aos hóspedes Não tem estacionamento mas, próximo tem um pago conveniado Recomendo nota 10
estacionamento ótimo	Excelente localização, um ótimo atendimento Café da manhã muito bom O apto triplo bem estruturado, dando privacidade aos hóspedes Não tem estacionamento mas, próximo tem um pago conveniado Recomendo nota 10
excelente quarto	Excelente qualidade preço pouco ha a mudar, excelente atendimento, dos melhores hoteis da cidade, as vistas do meu quarto tambem eram bastante razoaveis, pedi um taxi e chamaram me de imediato simplesmente fantastico
quarto fantastico	Excelente qualidade preço pouco ha a mudar, excelente atendimento, dos melhores hoteis da cidade, as vistas do meu quarto tambem eram bastante razoaveis, pedi um taxi e chamaram me de imediato simplesmente fantastico

Table 4.1: Table that shows some reviews and wrong associations due to the incorrect formatting of the text.

Next, was verified the occurrence of any type of points (!,?,.), commas or coordinating conjunctions (like the word 'mas'[14]) between the aspect and opinion words, and if they existed special care was needed to be taken.

[14]This word means 'but' in English.

The existence of points needs to be checked because *SyntaxNet* sometimes assumes that words with points between them are directly associated, which, after extensive analysis of 50 comments, was found as not being true in most of the cases, as can be seen at Table 4.2. So, this means that when it happened to have points between the aspect and opinion words they were considered non-associated.

Wrong associations	Comments
Quarto perto	Perto da ribeira e de atrações históricas. Quarto muito limpo e ótimo atendimento. O saguão (lobby) só e' frio porque o prédio e de pedra. Tudo muito bem preservado . Wi-fi grátis .O café da manha também e muito bom.
tudo muito bom	Perto da ribeira e de atrações históricas. Quarto muito limpo e ótimo atendimento. O saguão (lobby) só e' frio porque o prédio e de pedra. Tudo muito bem preservado . Wi-fi grátis .O café da manha também e muito bom.
pequeno-almoço péssimo	O pequeno-almoço era muito bom mas o serviço era péssimo.
quarto feia	Era muito feia a decoração, o quarto pequeno, só gostamos da comida.

Table 4.2: Wrong associations when appears points, commas and the word 'mas' between an aspect and an opinion word.

Contrarily of points, the commas and the coordinating conjunctions are more sensitive cases, and because of that a few more factors were needed to taking into account before guarantees that an aspect term is not associated with an opinion word. For example, in Table 4.2, the two last rows shows examples of that:

- The first case was corrected by identifying, not only, the word 'mas' between the aspect 'pequeno-almoço' and the opinion 'péssimo' but also another aspect word between them ('serviço'). If there was no different aspect word between them, they could be associated.
- The last one was corrected by identifying a comma between the two words ('quarto' and 'feia'), and also that the opinion word was in the female gender and the aspect was not.

If the found associations passed the below initial tests, another type of distance between the aspect's and opinion's words of these associations was verified taking into account the graph created previously. This distance was measured by using a *Networkx* function named *dijkstra_path*: Function that returns the shortest path from source to target in a weighted graph G (in this case between the opinion and aspect words).

Almost only the associations that had a length path equal to or less than 2 were considered and saved. There was only one special case, when they had length 3 and the term 'e'[15] or a verb word included in this path.

It should be noted that some adverbs of intensity plus prepositions had been also taken into account and, whenever these were directly associated with the opinion word, they joined the expression of the final association. Table 4.3 exposes some examples.

[15]It means 'and'.

Associations	Comments
Café da manhã muito bom	Café da manhã muito bom.
comida sem gosto	Comida sem gosto. Hotel bastante sujo e muitíssimo fúnebre.
Hotel bastante sujo	Comida sem gosto. Hotel bastante sujo e muitíssimo fúnebre.
Hotel muitíssimo funebre	Comida sem gosto. Hotel bastante sujo e muitíssimo fúnebre.

Table 4.3: Some comments and respective associations found with adverbs and prepositions included.

4.2.4 Classify associations found

To classify the associations found were used two types of predictions. One obtained from a neural network model and another from a *Python* module named *TextBlob*.

The neural network structure is practically the same as those used to classify comments in Portuguese and English (as visible in Figure 4.4). The actual changes occurred only at the input level. To classify the comments the training input and test was literally the comments, but in this case, the entries were associations. These associations used in this model were obtained from a dataset created using the algorithm previously explained in this section (a more detailed explanation of this creation is in the Chapter 3, Section 3.3).

To get better results was also needed one more LSTM and Dropout layers in the neural network model, but beyond these, extra layers did not improve significantly the model, they just increased the training time, therefore were discarded.

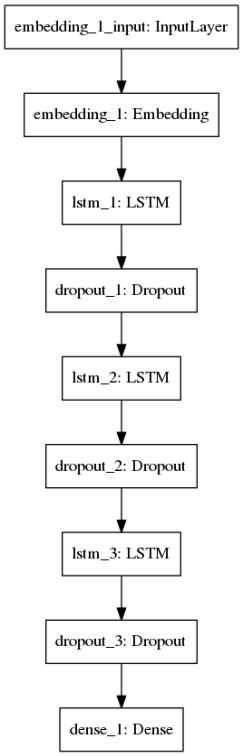


Figure 4.4: Neural network model to classify associations.

This model was trained and validated in three steps (all running 50 epochs of training and validation). In the first step were only considered 650 associations as inputs, in the second 700 and in the last one 723 examples.

The *TextBlob* module is a *Python* library for processing textual data. It provides an API that is able to do part-of-speech tagging, sentiment analysis, classification, translation, along with others [78].

In this thesis the functions used were the translation, Language Detection and the sentiment analyzer (named *PatternAnalyzer*). Firstly, the associations needed to be translated to English for a better result, after that was obtained the sentiment of the expressions throught the following equation:

$$sentiment = sentiment.polarity * sentiment.subjectivity \tag{4.5}$$

where the polarity and subjectivity were calculated by the *TextBlob* library. The result of this equation is a real number between -1 and 1. To turn this into five classes (from one to five) was created a *python* script (for example, a value between -1 and -0.6 belongs to class one).

Each time an association is classified, one of these two methods is chosen. This choice is made by the following manner:

- If the presented association was in the training neural network examples or the confidence[16] given by *TextBlob* was not 1.0 then the neural network prediction is used.
- If the association is unknown to the neural network model and *TextBlob* gives it 1.0 of confidence then the script *TextBlob* implementation is used.

This information had then to be permanently indexed. How was this done?

4.2.5 Index the information obtained

The comments have been analyzed and were found the associations present in them, being this done attraction by attraction (hotel or restaurant). The retrieved information was then temporarily secured in an array that contained the association, its score, the comment associated, the date of the comment, among other data that was found as pertinent.

Whenever all the comments of a particular attraction were analyzed, the information presented in the array was translated into a *JSON* format and permanently stored in an index using the *elasticSearch* tool.

Elasticsearch is a distributed, scalable RESTful real-time search and analytics engine that is able to search, analyze and explore the data saved (sometimes in ways that were never thought when starting a project). This tool exists because raw data saved on a hard drive is not very useful [79].

But why *elasticsearch* and not a regular database?

Traditionally, data was been stored in columns and rows in a relational database (like *SQL*). But using that, all the elasticity gained from using objects is lost because of the stiffness of a storage medium.

So why not save objects as objects? Instead of shaping an application around the limitations of spreadsheets (for example relational databases), it can preferably focus on using the

[16] In relation to the classification of a given association.

data. The flexibility of objects is returned to programmers - that is what *elasticsearch* allows [79].

An object is a language-specific, in-memory data structure. To send it across the web or save into a local store, is needed to be able to represent it in some standard format. *JSON* is a way of representing objects in human-readable text, and this tool only works over it [79].

In addition to this advantage, two another assets of *ElasticSearch* are its data fast retrieval and easy fitting with a web interface.

Back to the project. The information was saved into two different *elasticsearch* indexes, one for the hotels (*hotels-index*) and another to the restaurants (*restaurants-index*).

For each attraction, the average score of its general aspects, its highest and lowest scores, and how many positive and negative associations for that aspect occurred in the comments were also saved. An example of *JSON* information could be seen at Figure 4.5.

This saved information was then used by a web interface, being made queries to each one of the *Elasticsearch* indexes.


```
{
  "reviews": [
      {
        "date": "8 de junho de 2018",
        "text": "Comida muito boa mas serviço péssimo",
        "associations": {
          "serviço": [
            {
              "association": "serviço péssimo",
              "start": 234,
              "score": 1,
              "aspect": "serviço"
            }
          ],
          "comida": [
            {
              "association": "comida muito boa",
              "start": 31,
              "score": 5,
              "aspect": "comida"
            }
          ],
        },
        "title": "Podia ser melhor"
      }
  ],
  "rating": "4",
  "info": {
    "comida": {
      "media": 5,
      "negative associations": 0,
      "higher value": 5,
      "positive associations": 1,
      "lower value": 5
    },
    "serviço": {
      "media": 3,
      "negative associations": 1,
      "higher value": 1,
      "positive associations": 0,
      "lower value": 1
    }
  },
  "name": "Hotel Ficticio",
  "location": "Distrito de Aveiro"
}
```


Figure 4.5: Example of one *ElasticSearch* hotel index entry

Chapter 5

Results

5.1 Classification task

5.1.1 Evaluation

Before presenting the results for the classification's tasks, it is necessary to understand the metrics that are used to evaluate the model's performance.

In the field of machine learning and specifically for the problem of statistical classification are usually used the evaluation metrics that make use of four base definitions, called the confusion matrix, which measures the totality of outcomes:

- True Positive (known as TP): Examples predicted as one of multiple classes (for example in binary this means positive predictions) that matches the real class of them;
- False Positive (known as FP): Predictions of class examples that does not match the real class;
- True Negative (known as TN): Negative predictions that match the class's examples.
- False Negative (known as FN): Negative predictions that does not match the class's examples.

Taking into account this matrix were then calculated the evaluation metrics, and in this thesis were used four of them: *accuracy*, *precision*, *recall* and F_1 *score*.

The *accuracy* is the most common and simplest one that measures how often the model's predictions are correct:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.1}$$

This evaluation, although, is not enough to correctly evaluate these models tasks (multi-class and binary) because as seen at Chapter 3 the data is unbalanced and because of that this metric can not demonstrate how good the models were. Let's see an example to understand why this happens, if the output of nine reviews are class 4 and just two class 2 and the model predicts all the ten reviews to be in class 4 the accuracy will be 80 per cent. So this means that a model could easily achieve very high accuracy by always predicting the same class while completely failing other target classes.

So, the use of three more metrics was considered valuable. The *precision* that measures how often a class prediction is right, the *recall* that measures how often this class was predicted as being this class and F_1 *score* that is a combination of the last two metrics using harmonic averaging:

$$precision = \frac{TP}{TP + FP}, \tag{5.2}$$

$$recall = \frac{TP}{TP + FN}, \tag{5.3}$$

$$F_1 score = \frac{recall \times precision}{recall + precision} \times 2 \tag{5.4}$$

5.1.2 Multi-class classification task

In the multi-class classification task and for English dataset were calculated the *precision, recall and* f_1 *score* for all the five review classes (1-5) and the average of them obtained by the 2 final models (SVM and LSTM). The values presented in Tables 5.1 and 5.2 were obtained by testing the model for 5% of the total reviews (862 examples).

The one that got the best results was the LSTM neural network model, achieving **65%** of *precision*, **65%** of *recall* and **64%** of f_1 *score*, on average for all classes. The SVM model obtained, with their best parameters ($gamma = 512$, $C = 256$, and $kernel = poly$), 39% of *precision*, 40% of recall and 39% of f_1 *score*.

	precision	recall	f1-score	examples
class 1	0.29	0.42	0.34	57
class 2	0.33	0.50	0.40	72
class 3	0.42	0.35	0.38	99
class 4	0.34	0.33	0.33	250
class 5	0.56	0.38	0.45	384
avg/total	0.39	0.40	0.39	862

Table 5.1: Results for the English multi-class task using the final SVM model.

class	precision	recall	f1-score	examples
1	0.73	0.47	0.57	57
2	0.47	0.39	0.42	72
3	0.48	0.64	0.55	99
4	0.58	0.52	0.55	250
5	0.75	0.80	0.78	384
avg/total	0.65	0.65	0.65	862

Table 5.2: Results for the English multi-class task using the final neural network model.

For the accuracy, the displayed results in Table 5.3 were obtained by averaging the results achieved in each classes sub-sample. The best model for this metric was again the neural network model (66% against 45%).

In addition to the neural network model being better in all classification metrics, it was also much faster than SVM in training time. The SVM took 37 h 44 min to train and neural network only 9 h 38 min (about 4x less time training).

	Accuracy
Final SVM model	0.45
Final neural network model	**0.66**

Table 5.3: Accuracy results for the English multi-class task given by both final models.

For Portuguese dataset, the neural network model was again the best model. The results can be seen at Figures 5.4 and 5.5.

class	precision	recall	f1-score	examples
1	0.28	0.44	0.34	1 626
2	0.39	0.36	0.37	1 833
3	0.31	0.42	0.36	2 326
4	0.38	0.30	0.34	4 614
5	0.40	0.36	0.38	6 233
avg/total	0.35	0.42	0.36	16 632

Table 5.4: Results for the Portuguese multi-class task using the final SVM model.

class	precision	recall	f1-score	examples
1	0.54	0.66	0.60	1 626
2	0.48	0.37	0.41	1 833
3	0.56	0.56	0.56	2 326
4	0.60	0.57	0.59	4 614
5	0.77	0.80	0.78	6 233
avg/total	0.67	0.68	0.67	16 632

Table 5.5: Results for the Portuguese multi-class task using the final neural network model.

The average accuracy obtain by LSTM neural network for all classes was **78%**, SVM with their best parameters ($C = 1$ and $gamma = 0.5$), in its turn, obtained only 36%.

It was expected, as happened with LSTM model, that SVM also would increase its performance from English dataset to the Portuguese due to the number of comments be extra large, but this did not happen. This could have occurred because the data in English were cleanest and therefore more easily learned by SVM.

5.1.3 Binary classification task

For the binary classification task was only used the best multi-class classification model. As observed, this task fell on the LSTM neural network model (that was adapted for this type of classification, for more information see Chapter 4, Subsection 4.1.2).

	Accuracy	Precision	Recall	f1-score
English dataset	0.86	0.90	0.88	0.89
Portuguese dataset	**0.89**	**0.92**	**0.93**	**0.92**

Table 5.6: Results for the Portuguese and English binary classification task using the final neural network model.

As expected, the best accuracy was obtained for Portuguese dataset, this is due to the the difference of examples between the two datasets (20 000 for English dataset and 342 134 for Portuguese).

5.2 Identify and classify associations

In this section, the results under scrutiny are the associations found and their classification. Table 5.7 shows some of the results obtained using the algorithm described in Chapter 4, Section 4.2. These results provide confirmatory evidence that the method developed is able to find good associations and also predict their score with good correctness (evident in Table 5.8).

Association	Comment
Hotel excelente	Hotel excelente, comida muito boa e serviço espetacular.
Comida muito boa	Hotel excelente, comida muito boa e serviço espetacular.
Serviço espetacular	Hotel excelente, comida muito boa e serviço espetacular.
serviço não gostamos	Não gostamos do serviço, do quarto e da arrumação.
quarto não gostamos	Não gostamos do serviço, do quarto e da arrumação.
arrumação não gostamos	Não gostamos do serviço, do quarto e da arrumação.
espaço não gostamos	Não gostamos do espaço mas adoramos a paisagem
paisagem adoramos	Não gostamos do espaço mas adoramos a paisagem
Café da manhã sem gosto	Café da manhã sem gosto e fraco.
Café da manhã fraco	Café da manhã sem gosto e fraco.

Table 5.7: Some examples of associations detected by the developed algorithm (more information in the Chapter 4, Section 4.2.)

sentence	neural network	textBlob	Best choice model	real value
pessoal muito chato	1	1	1	1
cafe muito mau	1	1	1	1
cafe da manha muito mau	1	1	1	1
serviço horrivel	1	1	1	1
cozinha limpa	2	4	4	4
comida perfeita	5	5	5	5
zona muito linda	4	5	5	5
preço muito caro	1	2	1	1
local feio	3	1	1	1
decoração fantastica	4	4	4	4
atendimento nao gostamos	2	3	2	2
chef lugubre	2	3	2	2
pessoal pouco simpatico	2	3	2	2

Table 5.8: Some predictions obtained by using the neural network model for associations, using the *Textblob* library, and using the best choice between the two previous models and the real label of associations.

In relation to associations found, although most of the them are correct, there are still many incorrect ones, this is because *SyntaxNet* is not perfect and also because the Portuguese language is treacherous which leads to many comments being incorrectly written. The classification of associations sometimes are also incorrectly because neural network model had few training examples and *TextBlob* is not always correct.

5.3 Web Interface

This section represents the final visible results, where the information obtained, using the algorithm explained in Chapter 4, Section 4.2, is used by a web interface. This web interface was created using a high-level *Python* framework named *Django*, and below we can see some of the specific characteristics of it.

The Figure 5.1 shows the web page for hotels search. We can search for a hotel by putting is name and location or simply a hotel general aspect and its desired range score (for example, 1-5 as shown by the figure 5.2). After write all the desired parameters, it is only necessary to press the submit button.

Also note that, the top bar presented in this web page exists in all the web interface pages, and the restaurants web page search is exactly the same as this one.

After search for the desired hotels, will appear at maximum ten results divided by four pages. An example of an hotel retrieved by the search evidenced in Figure 5.1 is the Hotel Jardim. The initial information of this hotel presented in the resulting web page can be seen in Figure 5.3.

Figure 5.1: Web page to search for hotels.

Figure 5.2: Choose an existent hotel general aspect and the desired score between one and five.

The general information, that is evidenced in Figure 5.3, contains the general hotel aspects that appeared in at least one of the comments associated with its average score and the number of positive and negative associations. All of this information was obtained by taking into account every time the aspects related to a specific general aspect appeared in the comments

associated with an opinion word and their score resulting from the associations.

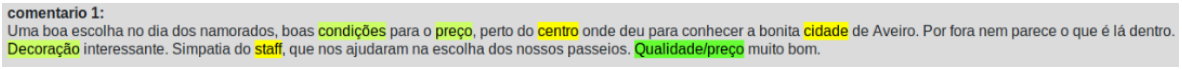
Hotel Jardim

Pontuação: 4
Localização: Distrito de Aveiro
Informação geral

Tipo	preço	serviço	outros	internet
Pontuação média	4	3	2	3
associações pos	25	75	24	7
associações neg	1	18	14	1

Figure 5.3: Example of the initial information retrieved for the Hotel Jardim.

The comments of this hotel appear then next to the initial information, as this hotel each of the establishments retrieved has the most three recent comments presented in section "comentários"[1]. To see all of their comments it is necessary to click the "Ver todos"[2] button. As evidenced by Figure 5.4, all of the comments have painted aspects. These aspects are those that belong to the associations found. For example, the most positive aspects (score five) that are associated with good opinion markers have a strong green color and the worst ones (score one) have an intense red color. The full variation can be seen in figure 5.5.



comentario 1:
Uma boa escolha no dia dos namorados, boas condições para o preço, perto do centro onde deu para conhecer a bonita cidade de Aveiro. Por fora nem parece o que é lá dentro. Decoração interessante. Simpatia do staff, que nos ajudaram na escolha dos nossos passeios. Qualidade/preço muito bom.

Figure 5.4: Example of one hotel comment.

Figure 5.5: Legend with color associate with its score.

In addition to the painted aspects, the comments also have associated the best and worst aspects presented in it (Figure 5.6).

To finish, all the information regarding the hotel Jardim can be seen in Figure 5.7, and all their comments at Figure 5.8.

[1]It means "comments".
[2]It means "See All".

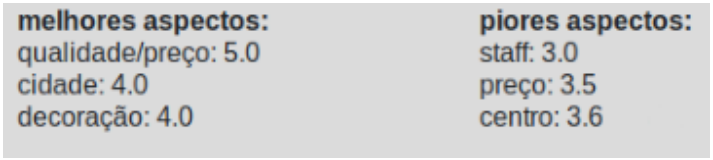


Figure 5.6: The best and worst aspects of one comment.

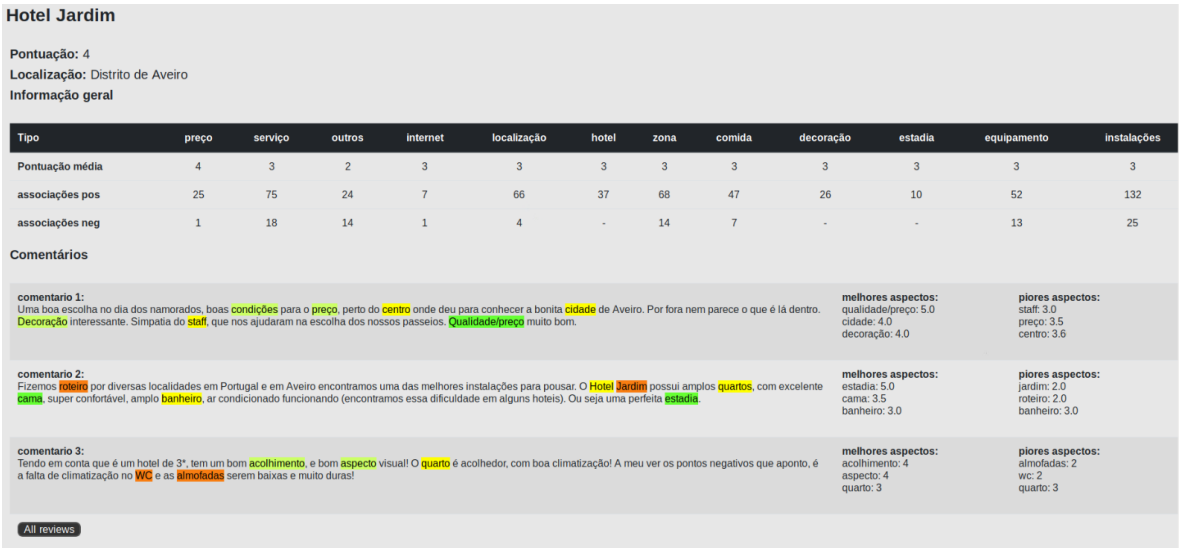


Figure 5.7: Web page that contains information about the Hotel Jardim.

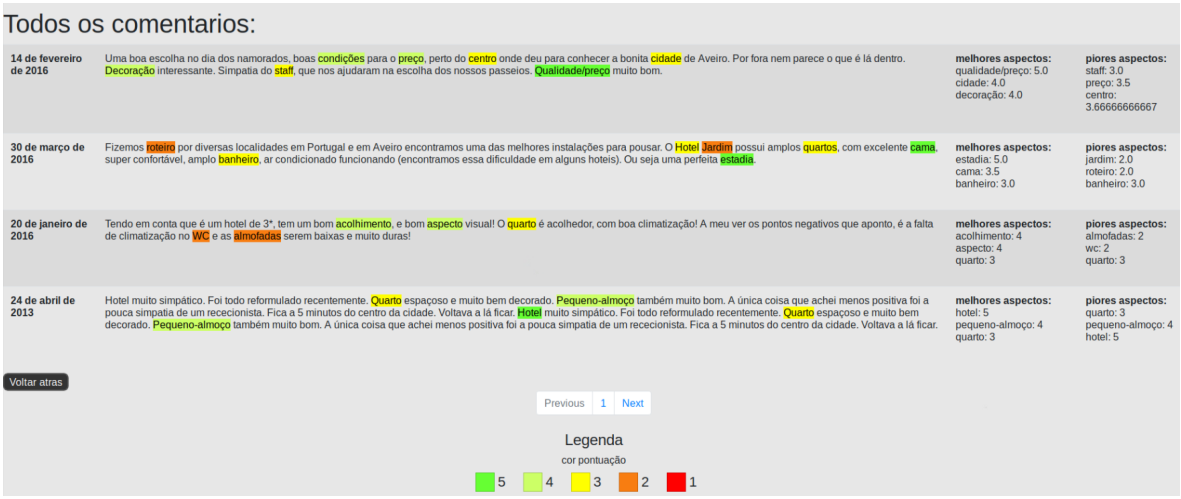


Figure 5.8: Web page with the first four reviews of a hotel, pagination allows to see all.

Chapter 6

Conclusions

Things like chatbots, machine learning tools, natural language processing, or sentiment analysis are applications of artificial intelligence that may one day profoundly change how we think about and transact in travel and local experiences[1].

In this thesis, the objectives under discussion were the development of a system that collects Portuguese reviews from *TripAdvisor's* hotels and restaurants, followed by their analysis and the creation of a web application for visualization of the information obtained. More specifically, the analysis of the comments consisted of two distinct tasks:

- The binary and multi-class classification of comments using a machine learning classifiers;
- The association of the attractions parameters (e.g. food, service) to feeling markers (e.g. good, bad), giving them a punctuation (from 1 to 5) by using lexical resources.

In order to achieve these desired goals, and before put hands at work, was done a state of the art research that was really useful to learn what tools were needed and how they worked.

The *Scrapy* framework was used to obtain about 400 thousand reviews from Portuguese *TripAdvisor*, and *Kaggle* provided a pre-created *TripAdivsor* dataset with reviews in English. After, this data was pre-processed before continue to the next stages and dictionaries were also created for the aspects of hotels and restaurants.

Then these reviews were classified using four classifiers for multi-class classification: Naive Bayes, Random Forest, SVM and Neural network LSTM. Of these four, the one that got the best results was the LSTM model, 66% of accuracy for English dataset and 78% for Portuguese. For binary classification was only use the LSTM model and it got 86% and 89% of accuracy, for English and Portuguese datasets, respectively.

The second task was accomplished by creating an algorithm, that operates using, for example, the *SyntaxNet* and *NetworkX* tools to find the associations between the attractions aspects and the feeling markers. After find this associations they were then classified by using a combination of a neural network model and the *TextBlob* framework. Finally, all the information obtained from this task was permanently saved using the *ElasticSearch* tool.

For all purposes, it is considered that the objectives have been fulfilled, being an important asset to support *TripAdvisor*, this works allows people to find information that is not currently present on this website.

[1] Gillian Tans

It should be noted that although the performance results of the classifiers could be eventually improved and most of the associations found are correct, there are still many incorrect ones, this is because *SyntaxNet* is not perfect and also because the Portuguese language is treacherous which leads to many comments being incorrectly written. And this is very difficult to correct because it took a long time to see all cases where bad associations occur.

So, after concluding the research and the intended tasks, it is consensual that there are some improvement avenues for further research.

In relation to the first task, the SVM and Random Forest models could improve results if used together with word embeddings (using, for example, the *Word2Vec* model), the LSTM model could be used in combination with a convolutional neural network to produce a greatest model, as demonstrated by Alec Yenter et al.[73].

For the second task, some of the misleading associations could be fixed, despite being a very time-consuming process (as discussed above) and could be created more training examples for the neural network LSTM model to improve the associations classification. Would also be interesting to find the aspects and opinions terms using machine learning models instead of dictionaries.

Bibliography

[1] Niels Brügger and Ralph Schroeder (2017). The Web as History: Using Web Archives to Understand the Past and the Present. UCL Press.

[2] Raffaele Filier (2015). What Makes Online Reviews Helpful? A Diagnosticity-Adoption Framework to Explain Informational and Normative Influences in e-WOM. Published in Journal of Business Research, 68(6), 1261-1270.

[3] *TripAdvisor* staff (2018). About TripAdvisor. Available at: `https://tripadvisor.mediaroom.com/US-about-us`. Accessed on: 29 May. 2018.

[4] Jayson DeMers (2015). Top 10 Reasons Your Brand Needs To Be On Twitter. Available at: `https://www.forbes.com/sites/jaysondemers/2015/07/07/top-10-reasons-your-brand-needs-to-be-on-twitter/#61b41ac4368b`. Accessed on: 25 Jan. 2018.

[5] Olga Kolodynska (2015). Word of Mouth Marketing: How to Get People Talking about Your Business. Available at: `https://www.livechatinc.com/blog/word-of-mouth/`. Accessed on: 25 Jan. 2018.

[6] Chaim Zins (2007). Conceptual approaches for defining data, information, and knowledge Published in Journal of the Association for Information Science and Technology, volume 58, pp 479-493.

[7] Rafael Almeida de Oliveira, and Renata Maria Arantes Baracho Porto (2016). EXTRACTING WEB DATA FROM TRIPADVISOR AS A SUPPORT FOR TOURISM INDICATORS DEVELOPMENT IN MINAS GERAIS.

[8] Madelyne Velasco, César San Lucas, Kevin Ortiz, Jose Vélez, and Carmen Vaca (2017). Secrets of Quito: Discovering a City Through TripAdvisor. Published in 2017 Fourth International Conference on eDemocracy & eGovernment (ICEDEG) Quito, Ecuador, April.

[9] Scrapy (Last commit at 10 Jan. 2018). Scrapy, a fast high-level web crawling & scraping framework for Python. Available at: `https://scrapy.org/`. Accessed on: 12 Jan. 2018.

[10] Eva Sánchez-Amboage, Verónica-Lucía MoraJácome, Ramiro-Leonardo Ramírez-Coronel, and Valentín-Alejandro Martínez-Fernández (2017). Ecuador's "four worlds" restaurants: Coast Region, Highlands, Amazon Region and Galápagos Islands on TripAdvisor.com. Published in 2017 12th Iberian Conference on Information Systems and Technologies (CISTI). Lisbon, Portugal, June.

[11] Maleerat Sodanil (2016). Multi-Language Sentiment Analysis for Hotel Reviews. Faculty of Information Technology, King Mongkut's University of Technology, North Bangkok, Thailand. Published in 2016 International Conference on Measurement Instrumentation and Electronics (ICMIE 2016).

[12] Francisco Villarroel-Ordenes, Stephan Ludwig, Dhruv Grewal, Ko de Ruyter, and Martin Wetzels (2016). Analyzing Online Reviews Through the Lens of Speech Act Theory: Implications for Consumer Sentiment Analysis. Published in Journal of Consumer Research, available online at: `http://jcr.oxfordjournals.org/`.

[13] Qurat Tul Ain, Mubashir Ali, Amna Riaz, Amna Noureen, Muhammad Kamran, Babar Hayat, and A. Rehman (2017). Sentiment analysis using deep learning techniques: a review. Published in Int J Adv Comput Sci Appl, Volume 8(6), pp 424.

[14] M. Day and C. Lee (2016). Deep Learning for Financial Sentiment Analysis on Finance News Providers, no. 1, pp. 11271134.

[15] Deng, L.; Yu, D. (2014). Deep Learning: Methods and Applications. Foundations and Trends in Signal Processing, pp 1–199.

[16] Bengio, Yoshua (2009). Learning Deep Architectures for AI. Foundations and Trends in Machine Learning, pp 1–127.

[17] Y. Zhang, M. J. Er, N. Wang, M. Pratama, and R. Venkatesan (2016). Sentiment Classification Using Comprehensive Attention Recurrent Models, pp. 15621569.

[18] Shiyang Yana, Jeremy S. Smith, Wenjin Lu, and Bailing Zhang (2017). CHAM: action recognition using convolutional hierarchical attention model.

[19] Jason Brownlee (2016). Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. Available at: `https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/`. Accessed on: 10 Jan. 2018.

[20] Suvro Banerjee (2018). An Introduction to Recurrent Neural Networks. Available at: `https://towardsdatascience.com/an-introduction-to-recurrent-neural-networks-72c97bf0912`. Accessed on: 10 April. 2018.

[21] Ron Kohavi, and Foster Provost (1998). Glossary of terms. Machine Learning. Published in Machine Learning (journal), pp 271-274.

[22] Vapnik, V.N. (1998). Statistical Learning Theory. Wiley, New-York.

[23] Boser, B.E., Guylon,I.M. and Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. Published in Proceedings of the 5th annual ACM workshop on Computational Learning Theory. ACM Press, pp 144-152.

[24] Jean-Philippe, V. (2002). A Tree kernel to analyse phylogenetic profiles. Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Kyoto, 611-0011, Japan.

[25] G. C. Cawley, and N. L. C. Talbot (2010). On Over-fitting in model selection and subsequent selection bias in performance evaluation. Published in Journal of Machine Learning Research, vol. 11, pp. 2079-2107, July.

[26] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau (2011). Sentiment analysis of Twitter data. Published in Proceeding LSM'11 Proceedings of the Workshop on Languages in Social Media, Pages 30-38. Portland, Oregon — June 23.

[27] Yujie Lu, Kotaro Sakamoto, Hideyuki Shibuki, and Tatsunori Mori (2017) Are Deep Learning Methods Better for Twitter Sentiment Analysis?. Graduate School of Environment and Information Sciences, Yokohama National University.

[28] Alec Go, Richa Bhayani, and Lei Huang (2009). Twitter sentiment classification using distant supervision. Published in CS224N Project Report (Stanford), pages 1–6.

[29] Yoon Kim (2014). Convolutional neural networks for sentence classification. Published in Proceedings of EMNLP 2014, pages 1746–1751.

[30] S. Wang, C. Manning (2012). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. Published in Proceedings of ACL.

[31] Xin Wang, Yuanchao Liu, Chengjie Sun, Baoxun Wang, and Xiaolong Wang (2015). Predicting polarities of tweets by composing word embeddings with long short-term memory. Published in Proceedings of ACL/IJCNLP 2015, pages 1343–353.

[32] Sara Rosenthal, Noura Farra, and Preslav Nakov (2017). SemEval-2017 Task 4: Sentiment Analysis in Twitter. Vancouver, Canada.

[33] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov (2016). SemEval-2016 Task 4: Sentiment Analysis in Twitter. San Diego, California, June 16-17.

[34] DeepDive. Distant supervision.
Available at: `http://deepdive.stanford.edu/distant_supervision`. Accessed on 26 Jan. 2018.

[35] Mickael Rouvier, and Benoit Favre (2016). SENSEI-LIF at SemEval-2016 Task 4: Polarity embedding fusion for robust sentiment analysis. Published in Proceedings of SemEval-2016, pages 202–208. San Diego, California, June 16-17.

[36] Nadeem N. Rather, Chintan O. Patel, and Sharib A. Khan (2017). Using Deep Learning Towards Biomedical Knowledge Discovery. Published in Online in MECS, April. Available online at `http://www.mecs-press.net/ijmsc`.

[37] Elisavet Palogiannidi, Athanasia Kolovou, Fenia Christopoulou, Filippos Kokkinos, Elias Iosif1, Nikolaos Malandrakis, Harris Papageorgiou, Shrikanth Narayanan, and Alexandros Potamianos (2016). Tweester at SemEval-2016 Task 4: Sentiment Analysis in Twitter using Semantic-Affective Model Adaptation. Published in Proceedings of SemEval-2016, pages 155–163. San Diego, California, June 16-17.

[38] Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif M. Mohammad (2014). NRC-Canada-2014: Detecting aspects and sentiment in customer reviews. Published in Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pp 437-442.

[39] Maria Pontiki, Haris Papageorgiou, Dimitrios Galanis, Ion Androutsopoulos, John Pavlopoulos, and Suresh Manandhar (2014). SemEval-2014 Task 4: Aspect Based Sentiment Analysis. Dublin, Ireland, August 23-24.

[40] Emmanuel Ahishakiye, Elisha Opiyo Omulo, Ruth Wario, and Ivan Niyonzima (2017). A Performance Analysis of Business Intelligence Techniques on Crime Prediction. Published in International Journal of Computer and Information Technology, Volume 06 – Issue 02, March.

[41] Naive Bayes Classifier. Available at: `http://www.statsoft.com/textbook/naive-bayes-classifier`. Accessed on: 28 Jan. 2018.

[42] Tin Kam Ho (1995). Random Decision Forests. Published in Proceedings of the 3rd International Conference on Document Analysis and Recognition, pp. 278–282. Montreal, QC, 14–16 August.

[43] Ramón Díaz-Uriarte, and Sara Alvarez de Andrés (2006). Gene selection and classification of microarray data using random forest. Published in BMC Bioinformatics 2006, January.

[44] Syncopation Software (2014). Instructional Decision Analysis Video: See Decision Analytics in Action!. Available at: `https://www.syncopation.com/instructional-decision-analysis-videos#decision_tree1`. Accessed on 20 Jan. 2018.

[45] Huan-Yuan Chen, and Hsin-Hsi Chen (2016). Implicit Polarity and Implicit Aspect Recognition in Opinion Mining. Published in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 20–25. Berlin, Germany, August 7-12.

[46] Dan Klein, and Christopher D. Manning (2003). Accurate unlexicalized parsing. Published in Proceedings of the 41st Meeting of the Association for Computational Linguistics, pages 423–430.

[47] T. Wilson, J. Wiebe, and P. Hoffmann (2005). Recognizing contextual polarity in phrase-level sentiment analysis. Published in Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing. Association for Computational Linguistics, pp. 347–354.

[48] Bin Lu, Myle Ott, Claire Cardie, and Benjamin K. Tsou (2011). Multi-aspect sentiment analysis with topic models. Published in Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, pp 81-88.

[49] Zhiqiang Toh, and Wenting Wang (2014). DLIREC: Aspect Term Extraction and Term Polarity Classification System. Published in Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pp 235-240.

[50] Slav Petrov (2016). Google Research Blog. Announcing SyntaxNet: The World's Most Accurate Parser Goes Open Source. Available at `https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html`. Accessed on 30 May, 2018.

[51] Josef Steinberger, Mohamed Ebrahim, Maud Ehrmann, Ali Hurriyetoglu, Mijail Kabadjov, Polina Lenkova, Ralf Steinberger, Hristo Tanev, Silvia Vázquez, and Vanni Zavarella (2012). Creating sentiment dictionaries via triangulation. Published in Decision Support Systems - Volume 53, Issue 4, pp 689-694.

[52] Walaa Medhat, Ahmed Hassan, and Hoda Korashy (2014). Sentiment analysis algorithms and applications: A survey. Published in Ain Shams Engineering Journal (2014) 5, pp 1093–1113.

[53] Andrea Esuli, and Fabrizio Sebastiani (2010). Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. Published in LREC, Volume 10 (2010), pp 2200-2204.

[54] Paula Carvalho, and Mario J. Silva (2015). Sentilex-PT: principais características e potencialidades. Published in Oslo Studies in Language 7(1), pp 425–438.

[55] C.J. Hutto, and Eric Gilbert (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Published in Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media.

[56] Lalit A. Patil, and S M. Kamalapur (2012). Improving web page clustering using Probabilistic Latent Semantic Analysis. Published in Proceedings published in International Journal of Computer Applications.

[57] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede (2011). Lexicon-based methods for sentiment analysis. Published in Computational linguistics, Volume 7(2), pp 267-307.

[58] Todor Mihaylov, Daniel Balchev, Yasen Kiprov, Ivan Koychev, and Preslav Nakov (2017). Large-scale goodness polarity lexicons for community question answering. Published in arXiv preprint arXiv:1707.06378.

[59] Saif M. Mohammad, and Peter D. Turney (2010). Emotions evoked by common words and phrases: Using Mechanical Turk to create an emotion lexicon. Published in Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text, pp 26-34. Association for Computational Linguistics.

[60] Saif M. Mohammad, and Peter D. Turney (2013). Crowdsourcing a word-emotion association lexicon. Published in Computational Intelligence Volume 29(3), pp 436-465.

[61] Thorsten Brants and Alex Franz (2006). Web 1t 5-gram version 1. Linguistic Data Consortium.

[62] Carlo Strapparava, and Alessandro Valitutti (2004). Wordnet affect: an affective extension of wordnet. Published in Lrec, Volume 4, pp 1083-1086.

[63] Minqing Hu, and Bing Liu (2004). Mining and Summarizing Customer Reviews. Published in Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 168-177.

[64] Giuseppe Castellucci, Simone Filice, Danilo Croce, Roberto Basili (2014). UNITOR: Aspect Based Sentiment Analysis with Structured Learning. Published in Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pp 761–767. Dublin, Ireland, August 23-24.

[65] Hossam S. Ibrahim, Sherif M. Abdou, and Mervat Gheith (2015). Idioms-proverbs lexicon for modern standard Arabic and colloquial sentiment analysis. Published in arXiv preprint arXiv:1506.01906.

[66] Mário J. Silva, Paula Carvalho, Carlos Costa, Luís Sarmento (2010). Automatic expansion of a social judgment lexicon for sentiment analysis. Published in silva2010automatic.

[67] Aric Hagberg, Dan Schult and Pieter Swart (2005). Networkx: Python software for the analysis of networks. Mathematical Modeling and Analysis, Los Alamos National Laboratory.

[68] Iñigo Jauregi Unanue, Ehsan Zare Borzeshi, and Massimo Piccardi (2017). Recurrent neural networks with specialized word embeddings for health-domain named-entity recognition. Published in Journal of biomedical informatics 76, pp. 102–109.

[69] Zhenzhen Li, Qun Zhang, Yang Liu, Dawei Feng, and Zhen Huang Recurrent neural networks with specialized word embedding for Chinese Clinical Named Entity Recognition.

[70] Tomas Mikolov, Wen-tau Yih, Geoffrey Zweig (2013). Linguistic regularities in continuous space word representations. Published in NAACL-HLT, pp. 746–751.

[71] Yoshua Bengio, Patrice Simard, and Paolo Frasconi (1994). Learning long-term dependencies with gradient descent is difficult. Published in IEEE transactions on neural networks 5.2, pp. 157-166.

[72] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi and Jurgen Schmidhuber (2001). Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-term Dependencies. Published in S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks.

[73] Alec Yenter, and Abhishek Verma (2017). Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. Published in Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017 IEEE 8th Annual.

[74] Shi Feng, Yang Wang, Liran Liu, Daling Wang, and Ge Yu (2018). Attention based hierarchical LSTM network for context-aware microblog sentiment classification. Pusblished in World Wide Web, pp 1-23.

[75] Guozheng Rao, Weihang Huang, Zhiyong Feng, and Qiong Cong (2018). LSTM with sentence representations for Document-level Sentiment Classification. Published in Neurocomputing.

[76] Diederik P Kingma, and Jimmy Ba (2014). Adam: A method for stochastic optimization. Published in arXiv preprint arXiv:1412.6980.

[77] Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016). Deep Learning. `http://www.deeplearningbook.org.` MIT Press.

[78] Steven Loria, Pete Keen, Matthew Honnibal, Roman Yankovsky, David Karesh, Evan Dempsey and others (2014). Textblob: simplified text processing. Pusblished in Secondary TextBlob: Simplified Text Processing.

[79] Clinton Gormley, and Zachary Tong (2015). Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine. O'Reilly Media, Inc.

[80] François Chollet and others (2015). Keras. Available at `https://keras.io`. Accessed on 10 May, 2018.

[81] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay (2011). Scikit-learn: Machine learning in Python. Published in Journal of machine learning research v12, pp 2825–2830.

[82] Gavin Hackeling (2014). Mastering Machine Learning with scikit-learn. Packt Publishing Ltd.