POLITECNICO DI MILANO
School of Industrual and Information Engineering
Master of Science in Computer Science and Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria

POLITECNICO
MILANO 1863

Twitter Sentiment Analysis

A comparison of a large subset of the techniques and services available to perform Sentiment Analysis applied to data from Twitter

Supervisor:
Prof. Licia Sbattella

Co-Supervisors:
Ing. Roberto Tedesco
Prof. Ernestina Menasalvas

Author:
Paolo Romeo, 919733

Academic Year 2020-2021

To my family, who has supported me during these years of intense studies.

Acknowledgements

I would like to sincerely thank my supervisors Ernestina Menasalvas, Roberto Tedesco and Licia Sbattella, who have demonstrated to be supportive during the discussion of ideas and methodologies to carry out the thesis.
Moreover, I would like to thank Gofore for allowing me to realize my graduation project at their company. It was an intense period and I learned a lot from both the professional and personal point of view. In particular I would like to thank Raul Martinez Lopez and Milla Siikanen for their availability and effort to make the process as smooth as possible and all the Madrid office for the hospitality and kindness shown during the internship.
Finally, I would like to thank the EIT community for the constant effort they are making to innovate the European education. It is thanks to their vision and commitment that I have been able to join the master and grow, personally and professionally.

Abstract

Many different models and services to perform Sentiment Analysis are available. It is often difficult to choose the right one for the use case of interest. This thesis analyses relevant techniques that have been successfully applied to classify sentiment polarity and it proposes a comparison of their performances based on experiments run on the dataset Sentiment140. Moreover, it proposes an analysis to understand when the models agree on the correct classification to highlight the margin of improvement that is possible to achieve in theory. Three main macro-categories of models are considered: traditional models based on mathematical theorems or intuitions (Naive Bayes, Support Vector Machine, Logistic Regression and Random Forest), neural models (ANN, CNN, Bi-LSTM and a hybrid approach) and classification services offered by top technology companies (AWS Comprehend, Google Natural Language API and Meaning Cloud). The tested models produced very similar performances, with the best model represented by Logistic Regression. Despite the potential of neural models and the advantages of ready-to-use services, traditional models proved to be the best trade-off and provided the best performances. Analyzing when the models agree, it was possible to observe that there is a subset of the dataset that is not correctly classified by any model, although in theory it is possible to achieve much better performances than those obtained by individual models.

Sommario

Sono disponibili numerosi modelli e servizi per eseguire la Sentiment Analysis. Spesso è difficile scegliere quello giusto per il caso d'uso di interesse. Questa tesi analizza un sottoinsieme delle tecniche più utilizzate che sono state applicate con successo per classificare la polarità del sentiment e propone un confronto delle loro prestazioni sulla base di esperimenti condotti sul dataset denominato Sentiment140. Viene inoltre proposta un'analisi per capire quando i modelli concordano sulla corretta classificazione, al fine di evidenziare il margine di miglioramento che sarebbe possibile ottenere in teoria. Vengono considerate tre principali macrocategorie di modelli: modelli tradizionali basati su teoremi o intuizioni matematiche (Naive Bayes, Support Vector Machine, Logistic Regression e Random Forest), modelli neurali (ANN, CNN, BiLSTM e un approccio ibrido) e servizi di classificazione offerti da alcune tra le aziende più attive nel settore (AWS Comprehend, Google Natural Language API e Meaning Cloud). I modelli testati hanno generato prestazioni molto simili, con il miglior modello rappresentato dalla Logistic Regression. Nonostante il potenziale dei modelli neurali e la facilità di utilizzo dei servizi pronti all'uso, i modelli tradizionali si sono dimostrati il miglior compromesso e hanno fornito le migliori prestazioni. Infine, si è osservato che esiste un sottoinsieme del dataset che non è classificato correttamente da nessun modello. In teoria, sarebbe però possibile ottenere performance molto migliori di quelle ottenute dai singoli modelli.

Contents

Chapter 1

Introduction

This first chapter introduces the concept of Sentiment Analysis and aims to raise awareness about its possible applications. It continues with the definition of the purpose and ends with the description of the structure of the document.

1.1 Motivations

Sentiment Analysis (SA) is a field of Natural Language Processing (NLP) that aims at extrapolating the sentiment embedded in a text. Most of the available text is unstructured [15] and it makes it hard for a machine to detect the proper polarity of the sentiment. It's a field that is growing fast under the interest of more and more researchers mostly because of the big achievements that have been obtained in recent times regarding computational power, thus enabling much more complex and time consuming computation that allows the use of neural models. According to [6] the market value of NLP will rise from 3 billions of US dollars in 2017 to more than 43 billions in 2025.

Sentiment Analysis is a classification task. Models that focus on polarity use as classes, for example, positive, negative and neutral. Models that focus on feelings and emotions may use angry, happy, sad, etc. If polarity precision is important to the application scenario, it can be meaningful to extend the categories to consider more shades. This approach is referred to as fine-grained sentiment analysis.

Moreover, depending on the goal of the application it is possible to define different granularity levels [10]. Document level Sentiment Analysis aims at detecting the sentiment of the whole text, allowing to exploit more data but generalizing on the content. An entity level approach allows to identify sentiment related to the entity that causes it. This is valuable, for example, in contexts like reviews of products because the product owner can understand what are the weaknesses and the strengths of the product itself and take countermeasures to align it to the expectations of the customers. Sentence level stands in the middle.

Thanks to social media, people are able to express their thoughts and feelings more openly than ever before. And they do. They express their thoughts about a product, a company, a service, political topics, science, events and so on and so forth.

For this reason Sentiment Analysis is extremely valuable for businesses. It allows to identify customer sentiment toward products, brands or services in online conversations and feedback.

Listening carefully to their customers, they are able to capture customers opinions and to react to them, tailoring a certain product or service to meet their needs and hence enhancing their value proposition and increasing customer satisfaction.

Proper Sentiment Analysis can help in predicting product sales performance, validating strategic and marketing decisions, brand monitoring, improving customer service and conducting market research.

Moreover, Sentiment Analysis can be effectively used to understand how the written communications are perceived and then to improve the tone of them. Businesses can use the service to learn the tone of their customers' communications and to respond to each customer appropriately, or to understand and improve their customer conversations.

Since Sentiment Analysis is a classification activity in its nature, after a correct encoding of the data it is possible to apply a multitude of techniques available to determine the output class. Freedom in the selection of different approaches exposes to choices that must be weighted on the basis of the knowledge of the approaches themselves and the dataset. A thorough knowledge of the different methodologies and the results that derive from their application is therefore fundamental in order to identify the best approach in each scenario.

1.2 Context

This master thesis is the final paper drawn up at the same time as an internship at the Gofore company. Gofore is a consultancy company that is mostly tackling digital transformation in the Finnish market and expanding their horizon to other European countries, like Spain and Germany. The work done may be of interest to those who deal with Sentiment Analysis and in particular Sentiment Analysis applied to messages from social platforms like Twitter or similar.
In writing this thesis it has been assumed that the reader has a basic understanding of common concepts in computer science, however, where possible, care has been taken in describing the critical steps and technologies.

The data considered to perform the analysis are taken from Twitter. Twitter is a micro-blogging and social networking service. Twitter users share their thoughts, news, real-time information and jokes in 280 characters of text or less, through messages called Tweets. The social network was launched in 2006 and nowadays it connects more than 330 million monthly active users. Among them, 152 million daily active users send 500 million tweets per day.[1] Users are not only individual people, businesses use Twitter to engage audiences within their sector; not only posting information about themselves but encouraging discussion, gathering useful feedback about products or services and offering helpful customer service. At the same time, customers also use Twitter as a quick and easy way to express opinions about a business, a product or a service.

1.3 Structure of the thesis

This thesis has been structured as follows. Chapter 2 introduces the research questions and narrows down the scope of the thesis.
In Chapter 3 the literature review is presented, to give an idea of what is the state of the art of Sentiment Analysis.
In Chapter 4 the publicly available dataset Sentiment140 is introduced and analyzed. It is the dataset used for the development and testing of the models.
In Chapter 5 the main techniques available for Sentiment Analysis are grouped into three distinct categories, determined by the nature of the approach and they are briefly described. Chapter 6 introduces some relevant information needed to carry out the experiments and presents the results of the application of the identified techniques to the dataset Sentiment140. Chapter 7 summarizes the findings of this study and presents some observations about the possibilities to enhance the performances of available techniques.

Chapter 2

Research questions and scope

This chapter introduces the questions that guided the research and defines the scope in relation to which the answers are more relevant.

2.1 Research questions

This thesis aims to identify some of the best models, available at the moment, to perform Sentiment Analysis on Twitter data and establish a structured comparison to highlight their advantages and disadvantages compared to the specific task of detecting the polarity of the Sentiment. Document level sentiment analysis is considered. However, since tweets are generally short in nature, most of them are made up of a single sentence. Hence, there is no substantial difference between sentence level and document level in this specific use case.
The first question can be summarized as follows:
Question 1: What is the best model to apply Sentiment Analysis on Twitter data?

The best model will be defined on the basis of the methodology introduced in

Chapter 4.

A second objective is to compare when the models agree on the correct classification, to understand if it is potentially possible to combine them in a new super-model that integrates their capabilities. How to create the model is not part of this research.
This second question can be defined as follows: Question 2: To what extent do the models identified as the answer to question 1 agree on the correct classification of the messages?

2.2 Scope

Restricting the scope in such a context is almost a must, since millions of messages are exchanged daily in numerous languages on Twitter and referring to the most varied topics.
Only Tweets belonging to the famous Sentiment140 dataset, strictly in English, are therefore considered. Further information regarding the dataset and the applied methodology are available in Chapter 4.

Chapter 3

State of the art

This chapter introduces related works that contribute to define the state of the art of Sentiment Analysis. First of all the main challenges that arise when dealing with Sentiment Analysis on Twitter data are presented. Then, some of the best available approaches that have been already developed and tested by the community of researchers and professionals to perform Sentiment Analysis are considered, distinguished by their nature: techniques focused on statistical methods, techniques based on neural networks and ready-to-use services offered by top technological companies.

3.1 Open challenges

In this section some of the most relevant open challenges that should be taken into consideration when selecting the right model to perform Sentiment Analysis on Twitter data are introduced. Some models can cope well with the presence of one of more of these aspects, while others cannot overcome these limitations because of the nature of the algorithms. In the next section 3.2 and in the chapter 5 the models and some insights related to when a model can be applied are detailed.

3.1.1 Emoticons and emojis

The information contained in a Twitter message is usually represented as textual content, though it's not the only possible way to express it.
Emoticons and emojis are effective ways to convey emotional status and intention of a message and they are recently gaining traction, becoming potentially useful to improve the quality of the models. [8] claims that many models provide low accuracy because they do not consider the presence and value of emoticons. Moreover, they present 3 classifications of sentiment polarity with different granularity. Eisentein noticed that the overwhelming majority of Twitter messages are not near the character limit and together with Pavalanathan published a paper [16] that stresses the rise of the emojis and their importance in the text. With words related to sentiments replaced by emoticons and emojis, they could become a key in the understanding of the sentiment.

3.1.2 Short informal messages

According to [18], short informal textual messages are limited in length, usually spanning one sentence or less. They tend to have many misspellings, slang terms, and shortened forms of words. They also have special markers such as hashtags that are used to facilitate search, but can also indicate a topic or sentiment. Tweets are usually extremely short. The limit is 280 characters, though most of them does not even hit the half of the limit. The Dataset chapter contains a detailed analysis of the length of the Tweets analysed.

3.1.3 Semantic, polysemy and sarcasm

Most of the approaches fail at capturing the semantic of the sentences because they just consider the words in it. The order of words is not considered at all. Moreover, the same words can have different meanings based on the context.
Embeddings are used to collect information about the semantic. Traditional Embedding fails to capture it and Naseem and Musial, in [4], propose a new model that combines 4 embeddings, each of them aiming at increasing the performances with respect to a specific threat. Context can completely change the meaning of the individual words in a sentence. It is for this reason that traditional word embeddings (word2vec, GloVe, fastText) fall short. They only have one representation per word, therefore they cannot capture how the meaning of each word can change based on surrounding context. The approaches used are Contextual ELMo, Word GloVe, PoS and Lexicon.
Sarcasm is another semantic-related problem. It is one one the most challenging issue in Sentiment Analysis.
Different speakers will tend to employ sarcasm regarding different subjects and, thus, sarcasm detection models ought to encode such speaker information. [13] introduces user embeddings to improve performances detecting sarcasm. A deep neural network is then fed with the additional information coming from traditional embeddings.

3.2 Available approaches

Many approaches have been developed and applied by researchers to address the use cases of Sentiment Analysis. In this section some of the best approaches that emerged from the literature review are listed. Models based on approaches that make explicit use of mathematical theorems and/or mathematical intuitions are firstly introduced and they are distinguished from models based on neural networks.

3.2.1 Traditional approaches

Before the rise of neural models, facilitated by the increased availability of computational power and data, Natural Language Processing and hence Sentiment Analysis was mainly based on statistical approaches. The simplest approach is the so-called "keyword-based", presented in [23].
[2] applies Decision Tree, Logistic Regression and Support Vector Machine (SVM) to perform Sentiment Analysis on the Twitter140 dataset. These are highly recurrent algorithms among the non neural approaches. [19] considers the same models and stresses the importance of pre-processing, proposing a structured way of doing it.

Yashaswini Hegde and S.K. Padma, in [12], propose the application of random forest technique to identify the polarity of the sentiment and test the performance of the same, comparing the results to the application of Naive Bayes to the same setting. It turned out that random forest performed better.

The aforementioned models, in the original setting, do not allow to capture the semantic of the sentences. They consider the words that compose the sentence but they completely ignore the order of the words. A particular succession of words can convey a totally different meaning than a different succession made up by the same set of words. This fact suggests to consider the context in which the word is placed in. In theory this is a highly expensive task from the computational point of view because to assess the effect of a word, all the other words should be considered each time. [9] introduces n-gram to face this issue. The principle on top of which the n-gram approach is built is to consider the text sentence as a set of sub-sentences made up by n contiguous words. It thus helps reducing the computation required, as well as the time needed for this operation. In this study, researchers apply n-grams to the input data before performing the analysis with Naive-Bayes and SVM and they show that it is possible to actually improve the accuracy of the models. In [23], the authors claim that using only bigrams as features is not useful because the feature space is very sparse. Combining unigram with bi-gram, they got a small improvement in all the models but SVM. Part of Speech tags are another way of adding information about the semantic and they are also considered in the same study. They turn to be not relevant in that context.

From the literature review, it emerges that the majority of the involved researchers agree on the fact that overall the best traditional models are Naive Bayes and SVM. Though, Logistic Regression and Random Forest in some cases give comparable performances. The cited models will be considered for the subsequent analyses.

3.2.2 Neural models

The neural models considered in this section are based on deep neural networks.

In [7], Paliwal, Kumar Khatri and M. Sharma apply the simplest form of artificial neural network (ANN) to run Sentiment Analysis. Despite in all the previous papers the results are based on accuracy only, here also precision and recall are included to measure the performance of the ANN. The authors finally state that such a type of neural network is very efficient in predicting the result with a high accuracy.

In [3] Jain & co. apply sentiment analysis on four small datasets, applying dropout to control overfitting. A hybrid approach made up by a Convolutional Neural Network (CNN) and Bidirectional Long Short Term Memory neural network (Bi-LSTM) is presented. The proposed method has been compared with various machine learning based methods and the experimental results show that the proposed method outperforms the existing methods over the considered datasets.

A similar solution is provided by a group of scientists lead by Mathieu Cliche during the SemEval 2017 (International Workshop on Semantic Evaluation). They won the competition using 10 CNN and 10 Bi-LSTM and applying soft-voting to ensemble the models, boosting the accuracy while reducing the variance [11]. The effect of applying a CNN is similar to using an N-gram approach in a non neural model.

3.2.3 Ready-to-use services

Recently, several top-tech companies are offering ready-to-use services that perform sentiment analysis applying state of the art techniques. Some of these services are free, or partially free up to a certain monthly use, some others are offered under pay-per-use plans at competitive prices. Here some of the main ones are reported and briefly described to allow getting insights about them. It's worth of note that sentiment models are defined for a particular language, hence each service works only with the specific set of languages it has been thought for. Ready-to-use services are extremely useful when limited expertise is available.

3.2.3.1 Amazon Comprehend

Amazon Comprehend[1] is a natural language processing service offered by Amazon Web Services. It is provisioned under a pay-per-use policy and it enables a broad range of applications that can analyze text exploiting machine learning techniques. Some of them are, for example, Entity Recognition, Sentiment Analysis, Syntax Analysis, Key Phrase Extraction, and Language Detection.

[1]https://aws.amazon.com/comprehend

The official description highlights that no experience with machine learning frameworks and/or models is required.

3.2.3.2 Google Cloud Natural Language API

The Natural Language API, offered by Google Cloud[2] , is extremely comprehensive for text analysis. It allows to identify the entities present in the text, it evaluates the polarity of the text at all three levels (document, sentence and entity level) and it carries out the PoS recognition enhanced by the morphological analysis of each identified part and the detection of dependencies between the words in the sentence. Each API supports different set of languages, though none of them works with Nordic languages. The Natural Language API supports 17 languages, including the major ones (English, Spanish and Chinese).

3.2.3.3 Meaning Cloud

The Meaning Cloud API[3] analyses the text to determine if it expresses a positive, negative or neutral sentiment. the local polarity of the different sentences in the text is identified and the relationship between them evaluated. It is possible to detect the polarity of user-defined entities and concepts, making the service a flexible tool applicable to any kind of scenarios. Beside the availability of several common languages, there exist a Nordic pack that includes Danish, Swedish, Norwegian and Finnish.

[2]https://cloud.google.com/natural-language

[3]https://www.meaningcloud.com/developer/documentation

Chapter 4

Materials and method

In this chapter the methodology, data and tools used to run the analysis are introduced to the reader in order to explain the main phases that have been carried out during the research and to facilitate replicability of the entire work.

4.1 Dataset

The dataset considered for the analysis is the famous sentiment140 dataset. It is a public dataset that contains 1,600,000 Tweets extracted using the twitter API [1]. Tweets are classified as positive or negative according to a semi-supervised classification technique called distant supervision. It uses rule based heuristics to produce labeled data. It thus allow to annotate a big corpus without manual labelling that it way more expensive. The process is described in detail in [23].

Each data entry has the following features:

- target: the polarity of the tweet (negative or positive)

[1]https://developer.twitter.com/en/docs/tweets/search/overview

- ids: an integer that represents the id of the tweet;
- date: the timestamp of the tweet;
- flag: the query that has been used to retrieve the Tweet with the Twitter API. If there is no query, then this value is NO_QUERY.
- user: the username of the user who published the message;
- text: the actual content of the tweet.

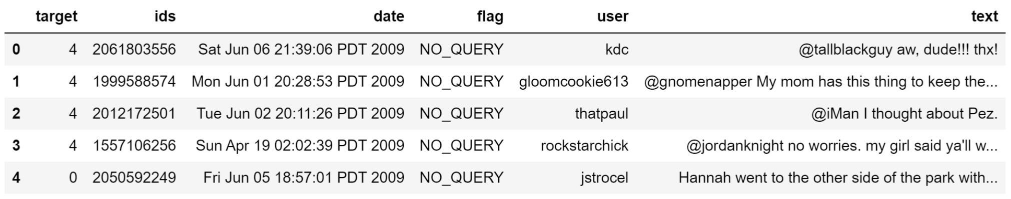
This is only a subset of the information that is obtained by using the Twitter API and it can be useful in other settings. Though, to accomplish the task of classifying the sentiment, it is enough to consider the fields target and text as they are the only ones that provide significant information for this purpose. It is not in the interest of this thesis to research further relationships between the features that could potentially improve the performance.

A descriptive analysis has been performed to better understand the dataset and get some insights about it. It is reported in the very next section along with some pre-processing tricks to reduce the noise of the data.

4.2 Analysis of the dataset and pre-processing

The descriptive analysis has been carried out at both character and word level, before and after the Pre-processing phase, to identify relevant insights about the length of the Tweets and the amount of information that is used to feed the models.

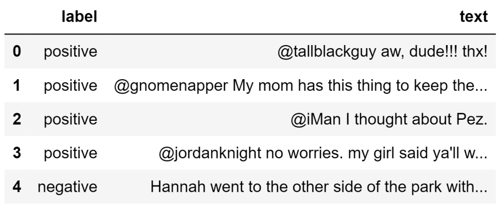
Figure 4.1 shows how the Sentiment140 dataset looks like, before any further operation performed on it.



	target	ids	date	flag	user	text
0	4	2061803556	Sat Jun 06 21:39:06 PDT 2009	NO_QUERY	kdc	@tallblackguy aw, dude!!! thx!
1	4	1999588574	Mon Jun 01 20:28:53 PDT 2009	NO_QUERY	gloomcookie613	@gnomenapper My mom has this thing to keep the...
2	4	2012172501	Tue Jun 02 20:11:26 PDT 2009	NO_QUERY	thatpaul	@iMan I thought about Pez.
3	4	1557106256	Sun Apr 19 02:02:39 PDT 2009	NO_QUERY	rockstarchick	@jordanknight no worries. my girl said ya'll w...
4	0	2050592249	Fri Jun 05 18:57:01 PDT 2009	NO_QUERY	jstrocel	Hannah went to the other side of the park with...

Figure 4.1: Original Sentiment140 dataset.

To the extent of this thesis there is no interest in looking for relationships among the features. For example, creating a history of the user and considering it to understand if the user is biased toward negative or positive writing could help increasing the performances of the model but it would require a much more complex processing of the available information. Hence, only target and text are kept for further analysis. Figure 4.2 shows the subset of feature further considered.



	label	text
0	positive	@tallblackguy aw, dude!!! thx!
1	positive	@gnomenapper My mom has this thing to keep the...
2	positive	@iMan I thought about Pez.
3	positive	@jordanknight no worries. my girl said ya'll w...
4	negative	Hannah went to the other side of the park with...

Figure 4.2: Restricted Sentiment140 dataset.

1600000 labelled messages are available, they are either positive or negative and the two classes are perfectly balanced as shown in Figure 4.3. This has been done of purpose by the creators of the dataset to limit the complexities of dealing with unbalanced classes and it, thus, simplifies the following phases.

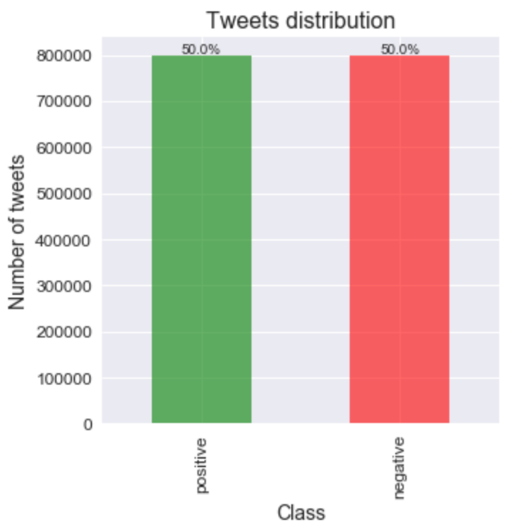


Figure 4.3: Distribution of the messages.

To understand the effect of pre-processing, a quantitative analysis has been run before taking any action on the corpus. Key statistics about the length of the Tweets are shown here in summary in Table 4.1. More detailed information is available in the appendix A.

Average number of characters	74
Longest tweet	374 characters
Shortest tweet	6 characters
Number of characters, quantile 0.99	141
Average number of words	13
Longest tweet	64 words
Shortest tweet	1 word
Number of words, quantile 0.99	28
Number of unique words	1350598

Table 4.1: Key statistics about the Tweets, before pre-processing

As it is evident, despite the maximum limit for a Tweet is 280 characters, 99%

of Tweets is made up by only half of the allowed characters.
Table 4.2 shows the exact same statistics, this time after pre-processing of the data. Hashtags, mentions and URLs are removed to reduce the noise of the dataset, since they do not contribute to bring information necessary for the final classification. At the same time, punctuation, numbers and stopwords have been removed. The complete set of stopwords, taken from the Natural Language Toolkit (NLTK) library, is available in the appendix B.

Average number of characters	40
Longest tweet	189 characters
Shortest tweet	0 characters
Number of characters, quantile 0.99	96
Average number of words	6
Longest tweet	50
Shortest tweet	0
Number of words, quantile 0.99	16
Number of unique words	249145

Table 4.2: Key statistics about the Tweets, after pre-processing

Removing hashtags and mentions, drastically reduced the number of unique words mostly because mentions refer to usernames that are unique by definition. At the same time, removing the stop words reduced the total number of words.
Quantitatively, the number of unique words in the corpus dropped by 81.55% while total words dropped by 48.48%. Dealing with the clean data enabled faster training and thus allowed to run multiple experiments despite the limited computational resources available.

4.3 Methodology

A precise flow have been followed to run the experiments, in order to avoid introducing bias due to unstructured executions. Starting from the raw dataset, data have been pre-processed removing hashtags, mentions, punctuation and stop-words, to reduce the uninformative data, thus reducing the overall size of data to manage, speeding up the experiments.

To use textual data for predictive modeling, the text must be transformed to be utilizable by the algorithms that are designed to work with either integers or real numbers. The transformation consists in two steps: tokenization and vectorization. Tokenization refers to the split of the text into words, or tokens, that represent the atomic unit of information. The text is so seen as a sequence of tokens and the next step, vectorization, maps tokens to a numerical representation.
So, before using the data they must be encoded in a proper way. Each encoding technique represents data in a different way and hence it can influence the performances of the models. Count vectorizer and tf-idf vectorizer have been identified to satisfy this need when dealing with traditional models. Likewise, for the neural models, data have been encoded with the TensorFlow vectorizer.
These techniques are shortly described below.

4.3.1 CountVectorizer

CountVectorizer is part of the library Scikit-learn [20]. It allows to convert a collection of text documents to a matrix of token counts. The result is a sparse representation of the counts using scipy.sparse.csr_matrix. The new representation of the data has a number of features equal to the vocabulary size found by analyzing the data. By the way the messages are composed by only a few words

with respect to the total number of tokens, thus, the use of sparse matrices allows to reduce a lot the space required to store it.
It is a simple yet basic approach that do not considers the order of the words in a sentence and the importance of each word.

Figure 4.4 shows how it works with a simple example.

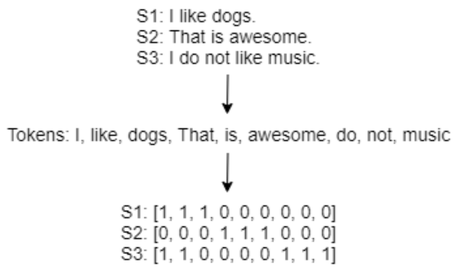

S1: I like dogs.
S2: That is awesome.
S3: I do not like music.
Tokens: I, like, dogs, That, is, awesome, do, not, music
S1: [1, 1, 1, 0, 0, 0, 0, 0, 0]
S2: [0, 0, 0, 1, 1, 1, 0, 0, 0]
S3: [1, 1, 0, 0, 0, 0, 1, 1, 1]

Figure 4.4: How CountVectorizer works.

4.3.2 Tf-idf vectorizer

Following the idea that a word that occurs many times is less informative in the encoded vectors than each other word that occurs less frequently, an alternative to the count vectorization is to calculate word frequencies. The most popular method that exploits this approach is called "Term Frequency - Inverse Document Frequency". Each word is assigned a value that is intended to reflect how important that word is to a document in a collection of documents [21]. This value, called tf-idf value, is the product of term frequency and inverse document frequency.
Defining n_t as the number of times the term t appears in a document and N_t as the total number of terms in the document, the term frequency of the term t is defined as follows:

$$TF(t) = \frac{n_t}{N_t} \tag{4.1}$$

In the same way, defining N_d at the total number of documents, or messages in the context of this thesis, and $n_{d,t}$ ad the number of documents that contain the term t, the inverse document frequency is defined as:

$$IDF(t) = log_e({}^{N_d}/_{n_{d,t}}) \tag{4.2}$$

The final value is nothing but the product of these two factors:

$$TF\text{-}IDF(t) = TF(t) \cdot IDF(t) \tag{4.3}$$

It increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

4.3.3 TensorFlow vectorization

The vectorization process started with the built-in Tokenizer available in TensorFlow. The process in made up of three steps. The first step creates the vocabulary of tokens to be used in the next step. This is the transformation of the sentences, seen as sequences of words, as sequences of tokens, replacing each word with its representation as token. Finally, since the neural networks need to work with inputs of equal size, every sequences of tokens id padded, adding zeros to the right, until the maximum specified length is reached. While allowing the processing of the data as integer numbers, it also reduces the overall memory to store the information since every word, composed by different characters, is mapped to a single integer. Figure 4.5 shows an example of this kind of vectorization.

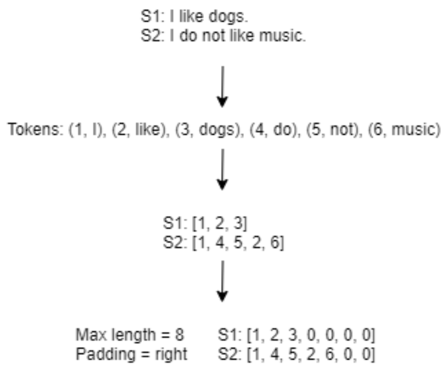
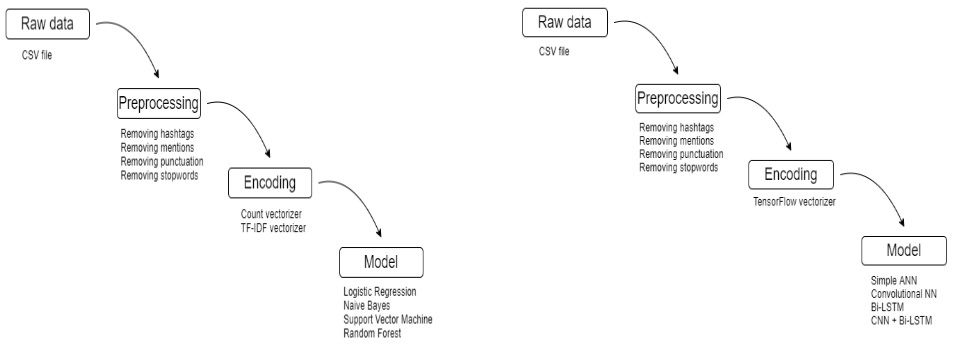

S1: I like dogs.
S2: I do not like music.
Tokens: (1, I), (2, like), (3, dogs), (4, do), (5, not), (6, music)
S1: [1, 2, 3]
S2: [1, 4, 5, 2, 6]
Max length = 8
Padding = right
S1: [1, 2, 3, 0, 0, 0, 0]
S2: [1, 4, 5, 2, 6, 0, 0]

Figure 4.5: Vectorization with TensorFlow.

Figure 4.6 schematizes the entire process from the raw data to the obtained models, highlighting the main phases and some key information that is crucial for each phase.



(a) Traditional models (b) Neural models

Figure 4.6: Overview of the main phases of the whole process

4.4 Metrics

Accuracy, precision and recall are common metrics to measure performances of classification models. To remind their definition it is useful to introduce the confusion matrix. It is a matrix that summarizes how many input elements have been correctly classified and how many have not. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. Classified elements fall into the following categories:

- True positives (**TP**): positive items correctly classified as positive;
- True negatives (**TN**): negative items correctly classified as negative;
- False positives (**FP**): negative items wrongly classified as positive;
- False negatives (**FN**): positive items wrongly classified as negative;

Figure 4.7 helps to visualize how they fit into the four categories. The performance metrics are defined with respect to these terms.

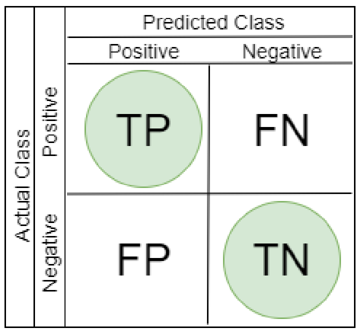


Figure 4.7: Confusion matrix for binary classification.

Accuracy represents the percentage of total items correctly classified.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

It is meaningful and easy to interpret when the two classes are balanced, like in the case of the Sentiment140 dataset.

Precision represents the percentage of items correctly identified as positive out of total items classified as positive.

$$Precision = \frac{TP}{TP + FP} \tag{4.5}$$

Recall, also called Sensitivity in binary classification, is the percentage of items correctly classified as positive out of total positives

$$Recall = \frac{TP}{TP + FN} \tag{4.6}$$

4.5 Tools

The Jupyter Notebook computational environment has been used to implement the models and run the experiments related to the performances achieved by all of them. The programming language has been Pyhton 3.7.
To complement the functionalities offered by the built-in Python libraries, Natural Language Toolkit (NLTK), Scikit-learn, Tensorflow and Keras have been used. With the last two libraries exclusively considered for the neural models.
The interactive setting of Jupyter notebooks turned to be extremely valuable and easy to use, especially when dealing with ready-to-use services.

Next chapter introduces the models that have been selected after the literature review.

Chapter 5

Modelling

In this chapter, the models selected for the comparison are described. Detailed information about the parameters used during the training are included, to facilitate the replication of the experiments.

5.1 Traditional approaches

Naive Bayes, Logistic Regression, Support Vector Machine and Random Forest are among the best non neural models for Sentiment Analysis according to the literature review that has been carried out. The scikit-learn library offers an implementation of these classifiers.

5.1.1 Naive Bayes

The Naive Bayes classifier is a probabilistic classifier based on Bayes' theorem and the assumption that features are independent of one another given some class [22]. The default scikit-learn implementation of Multinomial Naive Bayes is the one that is used for the experiments. According to [22], Naive Bayes is one of the most powerful algorithms for classification, and it works well even with millions of data entries, being fast to train.

5.1.2 Support Vector Machine

A Support vector Machine (SVM) is an efficient supervised machine learning algorithm used for classification. It is a non-probabilistic linear classifier that looks for the separating hyper-plane that maximizes the distance between the hyper-plane itself and the closest data point, namely margin. It performs very well with a limited amount of data and its effect increases with the increase in dimensional space [24].
The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples [20].
After some experiments run with different combinations of kernel and gamma, it has been decided to use a radial basis function as kernel function. Moreover, the value of the C parameter, that controls the cost of classification errors, has been set to 1. In general, a high value of C leads to a tighter margin, trying to avoid errors, that are heavily penalized and possibly leading to overfitting.

5.1.3 Random Forest

Random forest is an ensemble learning method that can be used for classification or regression. It works building a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. The version used for the experiments builds 100 trees, also known as estimators. Generally more estimators correspond to better performance and efficiency. It exploits the concept of information gain index [12].

5.2 Neural approaches

In this section approaches based on neural networks are introduced. TensorFlow and Keras have been used to write the models. Choices that are valid for all the implemented networks are presented here, while the specific Keras layers used to build the networks and the hyper-parameters, optimized through manual tuning are introduced in the corresponding subsections.

Since output classes are only two, this is a case of binary classification and it is possible to use binary cross-entropy as loss measure. Being, N the number of samples, w the weights of the neural network, y_i the target label and $p(y_i)$ the predicted label, depending on the weights of the network, the loss is defined as follows:

$$L(w) = 1/N \cdot \sum_{n=1}^{N} (y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(p(1 - y_i))) \tag{5.1}$$

Moreover, the optimizer chosen is the Adam optimizer available within Keras. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. According to [17], the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters". It has been used with default parameters.

5.2.1 Word Embeddings

The vectorial representation is the starting point for the final transformation that enables the neural networks to learn from the input data. It is referred to as embedding and it is basically a mapping of input data to a vector of real numbers. The purpose of word embeddings is to capture the semantic meaning, mapping it to a geometrical space. This is done by associating a numeric vector to every word in a dictionary, such that the distance between any two vectors would capture part of the semantic relationship between the two associated words. The geometric space formed by these vectors is called an embedding space. Ideally, in a good embedding space, words with similar semantic are placed close to each other in the space, thus it is possible to classify the overall sentiment of a sentence, analysing the embedding representation of the words contained in it [14]. Keras offers a simple way to generate the embedding space. It is done through the Embedding layer. It take as input the sequences of integers, that has been generated through the vectorization, and outputs the new representation exploited by following layers.

5.2.2 Simple Artificial Neural Network

The simplest neural approach identified after the literature review is a feed-forward neural network. It is an artificial neural network wherein connections between the neurons do not form any cycle. The considered version only exploits 1 hidden layer. Figure 5.1 represents graphically the layers of this model.

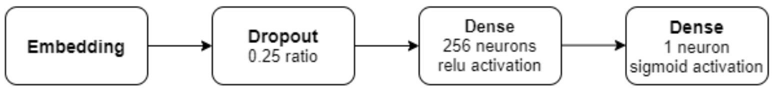


Figure 5.1: Layers of the artificial neural network.

5.2.3 Convolutional Neural Network

Figure 5.2 represents the layers of a Convolutional Neural Network(CNN). A CNN is another kind of feed forward neural network. The peculiarity of this approach is the application of two operations, convolution and pooling, that produce several changed representations of the input data, allowing to identify features in it. The obtained representations are a compression of the initial data, aiming at highlighting relevant information.

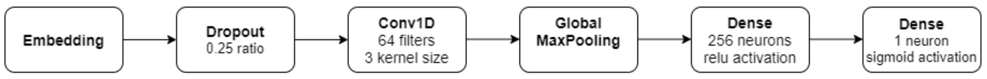


Figure 5.2: Layers of the convolutional neural network.

5.2.4 Bidirectional Long Short Term Memory

LSTM is a particular typology of recurrent neural network (RNN). The architecture of a RNN enables cyclic connections, allowing the network to reuse the output of neurons as input to other neurons backward, in combination with the current input data, thus establishing relationships among data. Unfortunately, when the gap between the relevant input data is large, the standard RNN fails at connecting the relevant information. Moreover, the standard RNN, is able to process only previous context, so new information can not influence the past directly. LSTM address the problem of connecting far information, introducing the ability to keep relevant information "in-memory". The Bidirectional property overcomes the other shortcoming allowing training in both time directions simultaneously, with separate hidden layers [5]. It actually uses two LSTMs, one LSTM acting in forward direction and the other one in backward direction.
Since it involves much more operations than the other neural models introduced above, it is way slower to train. Figure 5.3 shows the shows a representation of the layers of the BiLSTM.

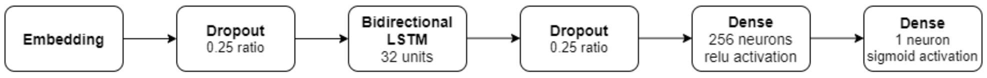
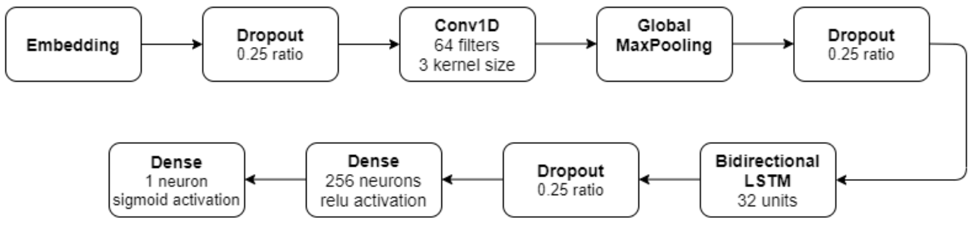


Figure 5.3: Layers of the Bi-LSTM neural network.

5.2.5 Hybrid Neural Network

Following the approach presented in [3] a hybrid solution considering both a CNN and Bi-LSTM has been built. Despite following its intuition, the models are combined in a different way than the one presented there. The novelty here, is given by the pipeline of the two processing operations. Instead of using an ensemble technique, the output of the CNN is provided as input to the Bi-LSTM in the attempt of catching more information about the semantic of the sentences. Figure 5.4 clarifies the layers involved in the complete model.



Embedding
Dropout 0.25 ratio
Conv1D 64 filters 3 kernel size
Global MaxPooling
Dropout 0.25 ratio
Bidirectional LSTM 32 units
Dropout 0.25 ratio
Dense 256 neurons relu activation
Dense 1 neuron sigmoid activation

Figure 5.4: Layers of the hybrid neural network.

5.3 Ready-to-use services

Google Cloud Natural Language API

Google Natural Language API follows a modular design that allows to get only the analysis there is interest in. It provides document and sentence level sentiment analysis, entity recognition, entity level sentiment analysis, and other text annotations independently accessible. The language of the text is automatically detected at run-time. If the language is not recognized as one of the few that are supported, an exception is raised. Supported language for Sentiment Analysis at document or sentence level, at the time of this writing are 17 and all the major ones are available. English, Japanese and Spanish only are support at an entity

level.
The structure of the response is clear and contains the sentiment score for the document and for each identified sentence. A magnitude value is associated to the sentiment score and it shows the intensity of that sentence. A “magnitude” is a number ranging from 0 to infinity. It represents the weight of sentiment expressed in the statement, regardless of being positive or negative. Longer blocks of text with heavily weighted statements have higher magnitude values. The “Sentiment score” is a real number and it ranges from -1 to +1. -1 is really bad to +1 being very good. Anything close to 0 is a neutral score. Table 5.1 is taken from the official documentation and shows some sample values and how to interpret them:

Sentiment	Sample Values
Clearly Positive	"score": 0.8, "magnitude": 3.0
Clearly Negative	"score": -0.6, "magnitude": 4.0
Neutral	"score": 0.1, "magnitude": 0.0
Mixed	"score": 0.0, "magnitude": 4.0

Table 5.1: How to interpret the results of a Google Natural Language API call

5.3.1 Amazon Comprehend

The API of interest for the Sentiment Analysis task is accessible through Boto, the Amazon Web Services (AWS) SDK for Python that enables easy access to the API as well as to low-level AWS services.
Comprehend estimates the likelihood of a message to belong to the classes Positive, Negative, Neutral or Mixed and outputs the one with the highest value. It allows 50k API calls per month for a few services, among with there is the Sentiment Analysis one.

5.3.2 Meaning Cloud

Meaning cloud offers a handy API that is available for free up to 20000 requests. Premium plans allow to release the limit. Though even with the free plan it is possible to have access to the full analysis and to gain evidence of the performances of the service. To run the Sentiment Analysis it is necessary to send a "SentimentRequest", specifying the text and its language, in this case English. The results are wrapped into a "SentimentResponse" that includes much more than the information needed for the purpose of this thesis. For example, entity level SA is performed within the same API call.
Restricting the information to what is meaningful for the analysis of accuracy, precision and recall, the only relevant fields are "score_tag" and "confidence". Score_tag is the label predicted by the model. It is one of the following values:

- P+: strong positive;
- P: positive;
- NEU: neutral;
- N: negative;
- N+: strong negative;
- NONE: without sentiment;

Confidence represents the confidence associated with the prediction. Its value is an integer number in the 0-100 range and it depends on the total information available to take the decision. Full documentation is available in the official website. [1].

[1]https://www.meaningcloud.com/developer/sentiment-analysis/doc/2.1/response

Chapter 6

Experiments, results and discussion

This chapter presents the experiments conducted on all the models introduced in the previous chapter. Accuracy, precision, recall and time needed for the computation to finish are presented here for each model along with some considerations on the results obtained. The workflow followed to run the experiments has been already presented in section 4.

6.1 Traditional models

This section presents the results of the experiments for the traditional models. To facilitate the analysis of the algorithms given the available computational power, a subset of 160k Tweets has been randomly selected from the original dataset that contains 1.6 million messages. Thus, the results should be considered as a proxy of the real performances over the entire dataset.
Models have been trained on 80% of the sample data and tested on the remaining

20%.
To investigate the potential effect of a different representation of the data, two round of tests have been proposed for these subset of models. The first one, encoding the data with the count vectorizer and the second one encoding the data with the tf-idf vectorizer.

6.1.1 Encoding with a count vectorizer

The results of the execution of the experiments after encoding the messages with the count vectorizer available in the sklearn library are presented in Table 6.1.

Model	Accuracy	Precision	Recall	Execution time
Naive Bayes	0.7550	0.7473	0.7710	0.64 s
Logistic Regression	0.7631	0.7761	0.7399	3.79 s
SVM	0.7282	0.7753	0.6430	1521.61 s
Random Forest	0.7482	0.7441	0.7568	332.93 s

Table 6.1: Performances of the traditional models, count vectorizer

6.1.2 Encoding with a tf-idf vectorizer

The same experiments, this time encoding with the tf-idf vectorizer offered by the sklearn library led to the results shown in Table 6.2.

Model	Accuracy	Precision	Recall	Execution time
Naive Bayes	0.7499	0.7408	0.7690	0.61 s
Logistic Regression	0.7664	0.7776	0.7466	1.65 s
SVM	0.7331	0.7525	0.6950	1926.74 s
Random Forest	0.7497	0.7512	0.7458	282.98 s

Table 6.2: Performances of the traditional models, tf-idf vectorizer

According to the values of the metrics, it emerges that the encoding of the data does not actually significantly affect the classification capabilities of the models. The models provide comparable performances, though Logistic Regression seems to ensure slightly better accuracy in this setting, requiring extremely low training time.
SVM required more than 3000x time than Naive Bayes that turned to be the fastest model.

6.1.3 Analysis of the agreement

Table 6.3 shows when the models agree on the correct classification. The goal of this analysis is to check how often the different models agree and to try to understand if there is margin to think about a possible combination of the models to improve the overall performances. The analysis is limited to the accuracy for the benefit of display clarity.

Metric	Tf-idf	Count
Total agreement	60.11%	58.28%
Majority agreement	72.13%	71.96%
At least two	80.23%	80.91%
At least one	87.44%	88.31%
Average accuracy	74.97%	74.86%
Maximum accuracy	76.64%	76.31%

Table 6.3: Comparison of agreement, traditional models

Despite a lower percentage of times when all the models agree on the correct class, applying a majority voting policy, the accuracy would be close to the average one obtained by considering models individually. Probably, the most interesting finding is that even being able to identify the right model for the right input and encoding the data in the luckiest way, there would be 11.69% of wrongly classified sentences because none of the models catches the right class.

6.2 Neural models

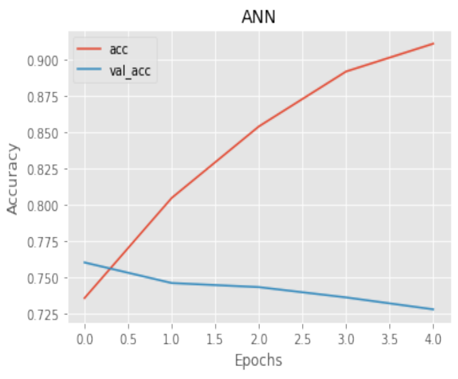
Performances of neural models are here shown. The split chosen for this category considered also a validation set that has been used to track validation error after each iteration of the training phase. Hence the split in this case was 60% training set, 20% validation set and 20% test set, training on 96k samples, validation and testing on 32k samples each.
Particular attention has been paid to ensure that the test data set is equivalent to that used for traditional models.

6.2.0.1 Simple ANN

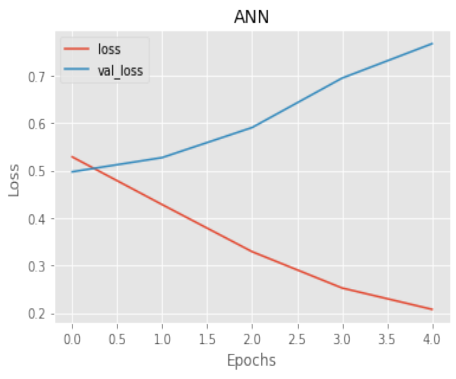
The results of running the simplest of the neural approaches are shown in summary in Table 6.4. The model has been trained for five epochs to understand the trend of the loss and the accuracy. Figure 6.1 shows how those two metrics behaved.

Simple ANN	
Accuracy	0.7248
Precision	0.7295
Recall	0.7229
Execution time	634.1 s
Time/epoch	126.82 s

Table 6.4: Performances of the simple ANN



(a) Accuracy



(b) Loss

Figure 6.1: Accuracy and Loss behaviour during the training phase of the ANN

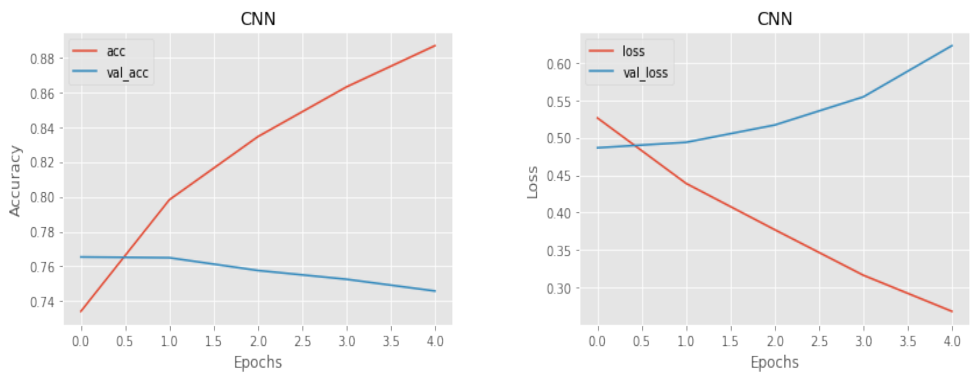
Despite the model is simple, it overfits just after two epochs, sign that the learning rate could be too big. Though, running the experiments with a smaller learning rate over a few epochs, gave a similar plot.

6.2.0.2 Convolutional NN

Table 6.5 shows the same experiments repeated with the CNN. The results turned to be slightly better. In particular, the increase of the precision is worth of mention. Figure 6.2 shows the behaviour of the Accuracy and the Loss of this model.

CNN	
Accuracy	0.7434
Precision	0.7638
Recall	0.7340
Execution time	732.51 s
Time/epoch	146.50 s

Table 6.5: Performances of the CNN



(a) Accuracy (b) Loss

Figure 6.2: Accuracy and Loss behaviour during the training phase of the CNN

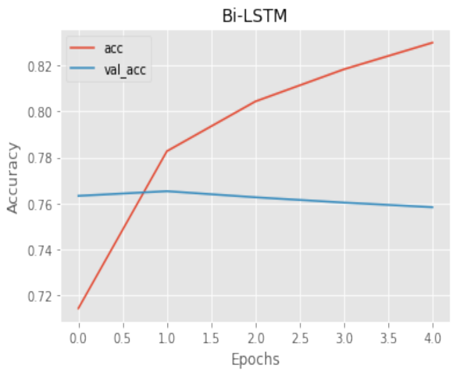
Even in this case, the model seems to overfit the training data after a few epochs, though the increase of the validation loss is still modest and the validation accuracy is steady.

6.2.0.3 Bidirectional LSTM

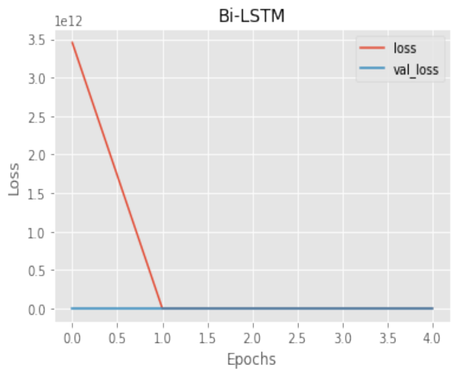
Bidirectional LSTM classified test samples better than the previous models. Though, it did not come for free: training time is way longer than the previous cases. Table 6.6 shows results of the training, while Figure 6.3 shows the behaviour of the Accuracy and Loss.

Hybrid	
Accuracy	0.7546
Precision	0.7439
Recall	0.7602
Execution time	3317.64 s
Time/epoch	663.53 s

Table 6.6: Performances of the bidirectional LSTM



(a) Accuracy



(b) Loss

Figure 6.3: Accuracy and Loss behaviour during the training phase of the Bi-LSTM

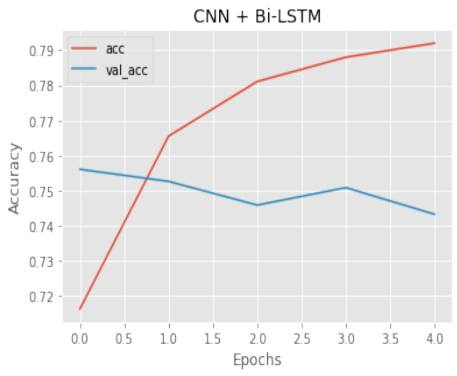
Analysing the plots, it emerges that initially the loss is huge but just after the first epoch it converges to values that are comparable to the previous models. Validation accuracy is steady, despite the constant rise of training accuracy.

6.2.0.4 Hybrid approach

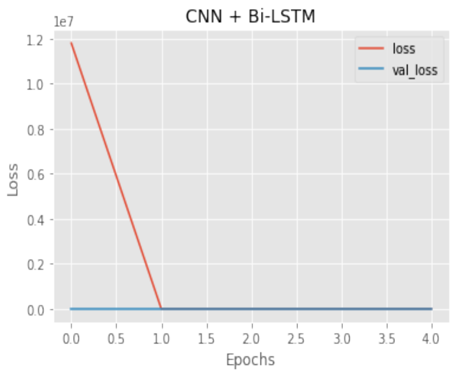
The model composed by a CNN and Bi-LSTM is more complex than the Bi-LSTM alone, though it requires less time to compute. This is due to the compression operated by the CNN that forces the Bi-LSTM to work with less data and thus to be faster. As Table 6.7 shows, its performances are slightly worse than the stand alone Bi-LSTM and it seems that is not worth making the model more complex, at least with this setting and hyper-parameters.

Bi-LSTM	
Accuracy	0.7439
Precision	0.7264
Recall	0.7528
Execution time	2255.36 s
Time/epoch	451.07 s

Table 6.7: Performances of the hybrid NN (CNN + Bi-LSTM)



(a) Accuracy



(b) Loss

Figure 6.4: Accuracy and Loss behaviour during the training phase of the hybrid NN (CNN + Bi-LSTM)

The behaviour of loss and accuracy, shown in Figure 6.4, is similar to the one of the Bi-LSTM, sign that the recurrent nature on the network prevails even in this case.

The performances of all the neural models are here summarized in Table 6.8, to allow a simpler visual comparison. The Bi-LSTM model provided the highest accuracy and it is highlighted in green. Table 6.9 shows it is also the model that required the longest training time.

Model	Accuracy	Precision	Recall
Simple NN	0.7248	0.7295	0.7229
CNN	0.7434	0.7638	0.7340
Bi-LSTM	0.7546	0.7439	0.7602
Hybrid	0.7439	0.7264	0.7528

Table 6.8: Summary of the performances of the neural models

Model	Execution time	Time per epoch
Simple NN	634.1 s	126.82 s
CNN	732.51 s	146.50 s
Bi-LSTM	3317.64 s	663.53 s
Hybrid	2255.36 s	451.07 s

Table 6.9: Summary of the time required by neural models

Overall, it seems that nature of the network is not a game changer. It is probably due to the size of the sample dataset that is considered. Moreover, with neural approaches the models overfit in a few epochs, even applying dropout with a relevant ratio. Also in this case the size and/or the nature of the dataset could be the main cause.

6.2.1 Analysis of the agreement

Table 6.10 shows the same analysis performed for the traditional models, this time applied to the neural ones. The analysis is limited to the accuracy for the benefit of display clarity.

Total agreement	56.71%
Majority agreement	71.30%
At least two	80.13%
At least one	88.53%
Average accuracy	74.17%
Maximum accuracy	75.46%

Table 6.10: Comparison of agreement, neural models

Similar conclusions to the traditional models case can be extrapolated here. There are no significant differences with respect to the agreement percentages of the traditional models.

6.3 Ready-to-use services

This section presents the results obtained by using the ready-to-use services widely described in the previous chapters. Overall, the services have been easy to use and interpret, with no machine learning expertise required. After running the API calls, some further analysis has been done to process the raw information and generate the confusion matrices and the key performance indicators accuracy, precision and recall.

In some cases, mapping between classes has been forced to allow having a comparison with the "handmade" models. For example, when both strongly positive and positive shades were available they have been mapped to positive. The same has been done for the negative counterpart. Messages predicted as Neutral or with no sentiment have not been considered for the performance calculation to comply with the initial labelling of the dataset that do not contain neutral values.

6.3.1 Meaning cloud

Meaning cloud had a limited free account that allowed to run 20k API calls. The following results has been obtained running 15k API calls since some calls were needed to set up the environment and get confidence with the results provided. Table 6.11 shows an overview of the results obtained.

Average time per call	0.9088 s
Average confidence	99.03%
Classified as Neutral	536 (3.57%)
Classified as "No sentiment"	4717 (31.45%)
Valid entries	10283
Accuracy	0.673
Precision	0.669
Recall	0.720

Table 6.11: Statistics of Meaning Cloud Sentiment Analysis API

31.45% of the analysed messages resulted to do not contain any sentiment according to this service. Given the high weight with respect to the total of messages, the performances are certainly optimistically biased.

6.3.2 Google Cloud Natural Language API

Google Cloud Natural Language API offered a much more flexible free tier that includes some credits to run a much more significant number of calls. Though, to simplify the comparison of the performances, even in this case 15k API calls are considered.

The sentiment is represented as a real value in [-1;+1], with -1 being the strongest negative and +1 the strongest positive. This scale naturally includes neutral

or mixed values that should not be considered in the count according to the chosen policy. Hence, messages with a predicted sentiment equal to zero are not considered in the performance evaluation.
Table 6.12 shows an overview of the results.

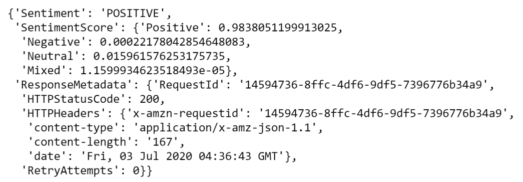
Average time per call	1.2401 s
Classified as Neutral	1788 (11.92%)
Valid entries	13212 (88.08%)
Accuracy	0.710
Precision	0.694
Recall	0.753

Table 6.12: Statistics of Google Cloud Natural Language API

Even in this case a quite big amount of information is lost to make the comparison possible. Anyway the performances of this model are comparable to the performances of the traditional and neural ones.

6.3.3 Amazon Comprehend

The Sentiment Analysis API within Amazon Comprehend classifies text as either Positive or Negative or Neutral or Mixed. It outputs both the final predicted class and an array of likelihood to all the possible labels. In the picture, a sample response is shown. In this case the message is classified as Positive with more than 98% of confidence.



```
{'Sentiment': 'POSITIVE',
 'SentimentScore': {'Positive': 0.9838051199913025,
  'Negative': 0.00022178042854648083,
  'Neutral': 0.015961576253175735,
  'Mixed': 1.1599934623518493e-05},
 'ResponseMetadata': {'RequestId': '14594736-8ffc-4df6-9df5-7396776b34a9',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '14594736-8ffc-4df6-9df5-7396776b34a9',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '167',
   'date': 'Fri, 03 Jul 2020 04:36:43 GMT'},
  'RetryAttempts': 0}}
```

Figure 6.5: Sample response to an API call of Amazon Comprehend.

Table 6.13 shows an overview of the results obtained testing on 15k samples.

Average time per call	1.3201 s
Classified as Mixed	911 (6.07%)
Classified as Neutral	6068 (40.45%)
Valid entries	8021 (53.47%)
Accuracy	0.769
Precision	0.750
Recall	0.790

Table 6.13: Statistics of Amazon Comprehend Sentiment Analysis API

Even in this case a quite big amount of information is lost to make the allow a fair comparison. Anyway the performances of this model are comparable to the performances of the traditional and neural ones, if samples classified as neutral are skipped.

6.3.4 Analysis of the agreement

Table 6.14 shows the same analysis performed for the traditional models, this time applied to the neural ones. Even in this case, the analysis is limited to the accuracy for the benefit of display clarity.
In this case, there is no need to skip neutral values, and values classified as neutral are considered as wrongly classified.

Total agreement	29.38%
Majority agreement	48.62%
At least one	71.73%

Table 6.14: Comparison of agreement, ready-to-use services

The potential accuracy, assuming it would be possible to select the correct prediction in every case, is quite high and it is likely that it is pessimistically biased since a lot of sentences have been classified as neutral by the considered services.

6.4 Final comparison

Table 6.15 and Table 6.16 summarize the performances of all the models and highlights the best model by typology of approach.

Model	**Accuracy**	**Precision**	**Recall**
Traditional models (count)			
Naive Bayes	0.7550	0.7473	0.7710
Logistic Regression	0.7631	0.7761	0.7399
Support Vector Machine	0.7282	0.7753	0.6430
Random Forest	0.7482	0.7441	0.7568
Traditional models (tf-idf)			
Naive Bayes	0.7499	0.7408	0.7690
Logistic Regression	0.7664	0.7776	0.7466
Support Vector Machine	0.7331	0.7525	0.6950
Random Forest	0.7497	0.7512	0.7458

Table 6.15: Summary of the performances of all the models - part 1

Finally, Table 6.17 shows the analysis of the agreement, extended to both traditional and neural models, to understand if there is a chance of improvement given the completely different nature of the two typologies of approaches. Since the performances of ready-to-use services have been assessed on a different subset of data, it is impossible to include them in this comparison. Moreover, only the prediction obtained after the tf-idf encoding for the traditional models is considered, to simplify the visualization and because the outcome of the two modalities

Model	**Accuracy**	**Precision**	**Recall**
Neural models			
Simple NN	0.7248	0.7295	0.7229
Convolutional NN	0.7434	0.7638	0.7340
Bidirectional LSTM	0.7546	0.7439	0.7602
CNN + Bi-LSTM	0.7439	0.7264	0.7528
Average metrics	0.7467	0.7524	0.7364
Best model	LR (tf-idf)	LR (tf-idf)	NB (count)
Ready-to-use services			
Google Cloud NLP API	0.710	0.694	0.753
Amazon Comprehend	0.768	0.750	0.790
Meaning Cloud	0.673	0.669	0.720

Table 6.16: Summary of the performances of all the models - part 2

is not significantly different.

8 different models are so compared: Naive Bayes, Logistic Regression, SVM, Random Forest, ANN, CNN, Bi-LSTM and the hybrid approach.

Total agreement	48.28%
At least seven	61.50%
At least six	69.10%
At least five	74.69%
At least four	79.17%
At least three	83.33%
At least two	87.79%
At least one	92.72%

Table 6.17: Comparison of agreement, traditional and neural approaches

The "irreducible error" drops to 7.28%, but selecting the right model would become more and more difficult in such a context. Applying a majority voting policy would produce the same average accuracy that is obtained using a single model. This approach would be hopefully more robust and less sensitive to noise but at the same time unreasonably complex for ordinary applications of Sentiment Analysis. Further research is needed to understand it is possible to reach that level of accuracy.

Chapter 7

Conclusions

This chapter summarizes the conclusions drawn from this research and provides suggestions for possible further research.

7.1 Results

It is convenient to start from the results deriving from data analysis and then continue with the models and their performances.
The analysis of the dataset provided interesting insights on the Sentiment140 dataset. First of all, although the maximum limit for a Tweet is 280 characters, 99% of the Tweets is made up of only half the allowed characters. In addition, the pre-processing carried out reduced the number of unique words in the corpus by 81.55 % while the total words decreased by 48.48%. Managing clean data allowed faster training and therefore allowed more experiments to be performed despite the limited computing resources available.
Regarding the models and their performances, it is appropriate to follow the same division that guided the drafting of the document. As for traditional models, the

model that produced the best accuracy, with an extremely low training time, is Logistic Regression, which managed to correctly classify 76.64% of the test data. Furthermore, despite some fluctuations in the exact values of the metrics, encoding with Count vectorizer or the tf-idf vectorizer did not significantly change the outcome.
Turning to neural models, Bi-LSTM is the best model with 75.46% accuracy. However, this model requires much more training time than all other models.
Finally, considering the services ready for use, Amazon Comprehend has proven to be the best with an accuracy of 76.8%. Unfortunately, however, it was not possible to carry out a highly structured analysis in this case because each service proposed a different scale of results, which was adapted to allow comparison. The use of ready-to-use services has not produced better results than the models implemented, although they can exploit more complex models, trained on significant amounts of data. On the other hand, they require no machine learning skills or computational power and are available at a very inexpensive price.
In general, the values of the accuracy obtained are more or less equivalent for each model. The oscillations are almost irrelevant and it should be taken into account that those values are the result of a fine tuning, even if only manual, to optimize the performance on the subset of samples extracted from the initial data set. So there seems to be no clear winner considering just the accuracy.
However, considering the skills required and the time needed for the calculation, the most promising models for this specific scenario seem to be Logistic Regression and Naive Bayes.

As for the analysis of when the models agree on the correct final decision, the results are more interesting.
For traditional and neural models considered separately, the results are similar. For about 12% of the test data, no model guesses the correct class, therefore it seems to be an insurmountable limit with current models. By applying a majority

voting policy, the accuracy is equal to the average obtained by considering the models individually.
Considering all the models at the same time the "irreducible error" drops to 7.28 % which leaves room for an accuracy that goes well beyond the capabilities of the models that currently existed. Again, by applying a majority voting policy, the accuracy is equal to the average. however, it is reasonable to imagine that such a model is more robust and less sensitive to noise. At the same time, it would be unreasonably complex for normal Sentiment Analysis applications. More research is needed to understand if that level of accuracy can be achieved.

7.2 Further steps

According to the scientific literature analyzed, it seems that the results obtained by traditional models are in line with the state of the art. As for neural models, they have the potential to outperform traditional models, although training requires significant expertise and resources. It is clear that the models considered are influenced by excessive adaptation and it is therefore probable that there is still room for improvement. A possible improvement for neural approaches consists in using already trained embeddings, instead of training them only on the single corpus considered. This measure could potentially overcome the limitation of access to restricted computational resources and therefore the need to consider a limited dataset. Furthermore, further research, including a more structured fine-tuning, is necessary to investigate the causes of overfitting and make the most of their potential.

Finally, by studying the agreement between models, it was shown "when" they agree. It would be appropriate to investigate whether there are clear reasons on "why" the models agree or not on the classification of the input data, to understand if considering a super model composed of various sub-models among those analyzed here, it is possible to improve its overall performance.

Appendix A

Dataset analysis

The tables below show some aggregated statistics about the length of the Tweets at a word and character level before applying any preprocessing techniques.

Average number of words	13
Longest tweet (words)	64
Shortest tweet (words)	1
Number of unique words	1350598
Total number of words	21081841
Number of words, quantile 0.25	7
Number of words, quantile 0.5	12
Number of words, quantile 0.75	19
Number of words, quantile 0.9	23
Number of words, quantile 0.99	28
Number of words, quantile 0.999	31

Table A.1: Full statistics about the Tweets, before pre-processing - word level

Average number of characters	74
Longest tweet (characters)	374
Shortest tweet (characters)	6
Number of characters, quantile 0.25	44
Number of characters, quantile 0.5	69
Number of characters, quantile 0.75	104
Number of characters, quantile 0.9	130
Number of characters, quantile 0.99	141
Number of characters, quantile 0.999	151

Table A.2: Full statistics about the Tweets, before pre-processing - character level

The tables below show the same aggregated statistics about the length of the Tweets at a word and character level after the preprocessing phase.

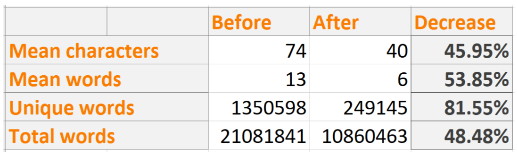
Average number of words	6
Longest tweet (words)	50
Shortest tweet (words)	0
Number of unique words	249145
Total number of words	10860463
Number of words, quantile 0.25	4
Number of words, quantile 0.5	6
Number of words, quantile 0.75	9
Number of words, quantile 0.9	12
Number of words, quantile 0.99	16
Number of words, quantile 0.999	19

Table A.3: Full statistics about the Tweets, after pre-processing - word level

Average number of characters	40
Longest tweet (characters)	189
Shortest tweet (characters)	0
Number of characters, quantile 0.25	22
Number of characters, quantile 0.5	37
Number of characters, quantile 0.75	56
Number of characters, quantile 0.9	73
Number of characters, quantile 0.99	96
Number of characters, quantile 0.999	110

Table A.4: Full statistics about the Tweets, after pre-processing - character level

The quantitative effect of pre-processing is shown in the following table.



	Before	After	Decrease
Mean characters	74	40	45.95%
Mean words	13	6	53.85%
Unique words	1350598	249145	81.55%
Total words	21081841	10860463	48.48%

Figure A.1: Effect of the pre-processing.

Finally, here is a cloud word representation of the corpus, after the pre-processing.

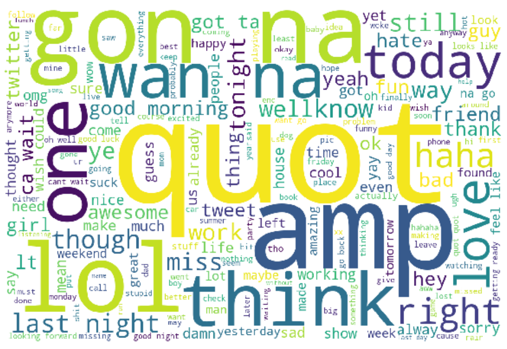


Figure A.2: Cloud word representation of the corpus after pre-processing.

Appendix B

List of stop words

0 - shouldn	36 - themselves	72 - weren	108 - so	144 - theirs
1 - we	37 - now	73 - about	109 - don’t	145 - again
2 - of	38 - each	74 - hasn’t	110 - during	146 - can
3 - am	39 - shan	75 - were	111 - he	147 - been
4 - if	40 - me	76 - shouldn’t	112 - by	148 - wasn
5 - through	41 - them	77 - a	113 - above	149 - up
6 - too	42 - both	78 - does	114 - t	150 - whom
7 - o	43 - it’s	79 - needn	115 - couldn’t	151 - y
8 - or	44 - ll	80 - you’re	116 - be	152 - why
9 - such	45 - won’t	81 - the	117 - few	153 - m
10 - more	46 - didn	82 - him	118 - should’ve	154 - itself
11 - below	47 - don	83 - its	119 - they	155 - i
12 - mustn	48 - wouldn’t	84 - mightn	120 - hadn’t	156 - some

Table B.1: Full list of stopwords removed from the corpus, part 1

13 - she's	49 - do	85 - didn't	121 - shan't	157 - wasn't
14 - doesn	50 - who	86 - she	122 - yourself	158 - aren
15 - at	51 - own	87 - these	123 - being	159 - couldn
16 - you'll	52 - isn't	88 - hadn	124 - you	160 - haven
17 - and	53 - then	89 - needn't	125 - no	161 - after
18 - not	54 - ours	90 - hers	126 - had	162 - doing
19 - are	55 - ourselves	91 - re	127 - ain	163 - into
20 - has	56 - very	92 - ma	128 - as	164 - just
21 - but	57 - all	93 - which	129 - their	165 - herself
22 - in	58 - that	94 - only	130 - here	166 - yours
23 - nor	59 - any	95 - s	131 - myself	167 - how
24 - same	60 - against	96 - down	132 - on	168 - mustn't
25 - isn	61 - haven't	97 - over	133 - for	169 - while
26 - that'll	62 - yourselves	98 - weren't	134 - himself	170 - to
27 - where	63 - aren't	99 - there	135 - out	171 - wouldn
28 - did	64 - is	100 - will	136 - hasn	172 - other
29 - d	65 - mightn't	101 - it	137 - with	173 - her
30 - you'd	66 - our	102 - because	138 - between	174 - most
31 - you've	67 - those	103 - once	139 - than	175 - this
32 - when	68 - an	104 - doesn't	140 - his	176 - should
33 - having	69 - further	105 - under	141 - have	177 - until
34 - before	70 - from	106 - off	142 - my	178 - what
35 - was	71 - ve	107 - won	143 - your	-

Table B.2: Full list of stopwords removed from the corpus, part 2

Acronyms

ANN Artificial Neural Network. 11

API Application Programming Interface. 34

AWS Amazon Web Services. 33

Bi-LSTM Bidirectional Long Short Term Memory. 11

CNN Convolutional Neural Network. 11

NLP Natural Language Processing. 1

NLTK Natural Language Toolkit. 19

RNN Recurrent Neural Network. 31

SA Sentiment Analysis. 1

SDK Software Development Kit. 33

SVM Support Vector Machine. 10

URL Uniform Resource Locator. 19

Glossary

Natural Language Processing Subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human languages. 1

Natural Language Toolkit Suite of libraries and programs for symbolic and statistical analysis in the field of Natural Language Processing. 19

Pre-processing Processing phase including cleaning, normalization, transformation, feature extraction and selection. 16

Sentiment A personal positive, neutral or negative feeling. 5

Tweets Messages exchanged on the micro-blogging platform Twitter. Their maximum length is 280 characters. 4

List of Figures

List of Tables

Bibliography

[1] J. Clement. *Twitter - Statistics & Facts.* [Online; accessed 06-April-2020]. Feb. 2020. URL: https://www.statista.com/topics/737/twitter/.

[2] A. Gupta et al. "Sentiment Analysis of Twitter Posts using Machine Learning Algorithms." In: *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom).* Mar. 2019, pp. 980–983. DOI: https://ieeexplore.ieee.org/document/8991365.

[3] D. Jain, A. Garg, and M. Saraswat. "Sentiment Analysis using Few Short Learning." In: *2019 Fifth International Conference on Image Information Processing (ICIIP).* Nov. 2019, pp. 102–107. DOI: 10.1109/ICIIP47207.2019.8985855.

[4] U. Naseem and K. Musial. "DICE: Deep Intelligent Contextual Embedding for Twitter Sentiment Analysis." In: *2019 International Conference on Document Analysis and Recognition (ICDAR).* Sept. 2019, pp. 953–958. DOI: 10.1109/ICDAR.2019.00157.

[5] Yong Yu et al. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures." In: *Neural Computation* 31.7 (2019). PMID: 31113301, pp. 1235–1270. DOI: 10.1162/neco_a_01199. URL: https://doi.org/10.1162/neco_a_01199.

[6] Shanhong Liu. *Revenues from the natural language processing (NLP) market worldwide from 2017 to 2025.* [Online; accessed 13-April-2020]. Dec.

2018. URL: https://www.statista.com/statistics/607891/worldwide-natural-language-processing-market-revenues/.

[7] S. Paliwal, S. Kumar Khatri, and M. Sharma. “Sentiment Analysis and Prediction Using Neural Networks.” In: *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. 2018, pp. 1035–1042.

[8] K. Utsu, J. Saito, and O. Uchida. “Sentiment Polarity Estimation of Emoticons by Polarity Scoring of Character Components.” In: *2018 IEEE Region Ten Symposium (Tensymp)*. Aug. 2018, pp. 237–242. DOI: 10.1109/TENCONSpring.2018.8691984.

[9] S. Wankhede et al. “Data Preprocessing for Efficient Sentimental Analysis.” In: *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. Apr. 2018, pp. 723–726. DOI: 10.1109/ICICCT.2018.8473277.

[10] P. Balaji, O. Nagaraju, and D. Haritha. “Levels of sentiment analysis and its challenges: A literature review.” In: *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*. Mar. 2017, pp. 436–439.

[11] Mathieu Cliche. *BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs*. 2017. arXiv: 1704.06125 [cs.CL].

[12] Y. Hegde and S. Padma. “Sentiment analysis using random forest ensemble for mobile product reviews in kannada.” In: *IEEE 7th International Advance Computing Conference (IACC)* (2017), pp. 777–782.

[13] Silvio Amir et al. *Modelling Context with User Embeddings for Sarcasm Detection in Social Media*. 2016. arXiv: 1607.00976 [cs.CL].

[14] Francois Chollet. *Using pre-trained word embeddings in a Keras model*. [Online; accessed 04-July-2020]. July 2016. URL: https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html.

[15] Christie Schneider. *The biggest data challenges that you might not even know you have.* [Online; accessed 06-April-2020]. May 2016. URL: `https://www.ibm.com/blogs/watson/2016/05/biggest-data-challenges-might-not-even-know/`.

[16] Umashanthi Pavalanathan and Jacob Eisenstein. *Emoticons vs. Emojis on Twitter: A Causal Inference Approach.* 2015. arXiv: `1510.08480 [cs.CL]`.

[17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2014. arXiv: `1412.6980 [cs.LG]`.

[18] Svetlana Kiritchenko, Xiaodan Zhu, and Saif Mohammad. "Sentiment Analysis of Short Informal Text." In: *The Journal of Artificial Intelligence Research (JAIR)* 50 (Aug. 2014). DOI: `10.1613/jair.4272`.

[19] R. de Groot. "Data Mining for Tweet Sentiment Classification." Utrecht University, 2012. DOI: `https://dspace.library.uu.nl/handle/1874/253766`.

[20] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[21] A. Rajaraman and J.D. Ullman. *Mining of Massive Datasets.* 2011, pp. 1–17. ISBN: ISBN 978-1-139-05845-2. DOI: `10.1017/CBO9781139058452.002`.

[22] Y. Zhang Y. Ji S. Yu. "A novel naive bayes model: Packaged hidden naive bayes." In: *6th IEEE Joint International Information Technology and Artificial Intelligence Conference* (2011), pp. 484–487.

[23] Alec Go, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." In: *Processing* 150 (Jan. 2009).

[24] I. Tsochantaridis et al. "Support vector machine learning for interdependent and structured output spaces." In: *Proceedings of the twenty-first international conference on Machine leaning, ACM* (2004), p. 104.