Giovanni Paolo Santos de Carvalho

Supervised Learning using Sparse and Dense Feature Representations: A Comparative study in Sentiment Analysis

Recife, Brazil

August - 2017

Giovanni Paolo Santos de Carvalho

Supervised Learning using Sparse and Dense Feature Representations: A Comparative study in Sentiment Analysis

Trabalho de Conclusão de Curso apresentado ao Programa de Bacharelado em Ciência da Computação do Departamento de Estatística e Informática da Universidade Federal Rural de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Estatística e Informática

Programa de Graduação

Supervisor: Rinaldo José de Lima

Recife, Brazil

August - 2017

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por Giovanni Paolo Santos de Carvalho às 15 horas do dia 28 de agosto de 2017, no Auditório do CEAGRI-02 – Sala 07, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado **Supervised Learning using Sparse and Dense Feature Representations: A Comparative study in Sentiment Analysis**, orientado por Rinaldo José de Lima e aprovado pela seguinte banca examinadora:

Rinaldo José de Lima
DEINFO/UFRPE

Rafael Ferreira Leite de Mello
DEINFO/UFRPE

George Gomes Cabral
DEINFO/UFRPE

Acknowledgements

I would like to thank my friends and family who always supported me throughout the difficult times. My mother, Renilda Santos, with her love and attention, and my father, Jânio Carvalho, with his words of wisdom.

I am grateful for my advisor, Rinaldo Lima, for all his help and lessons he has provided me. His assistance was essential to make this work possible.

I am also grateful for my colleagues and professors who were a part of my journey at Universidade Federal Rural de Pernambuco, and for all the life-changing experiences they have provided me.

Finally, I would like to express my sincerest love for my girlfriend, Gleyciane Alcântara, for the moments of joy we have shared together, for always believing in me and helping me find strength in my weakest moments.

Resumo

Nos últimos anos tem ocorrido um aumento exponencial na quantidade de conteúdo disponível publicamente na Web. Uma grande parte desse conteúdo está presente em forma de textos em Linguagem Natural, que é inerentemente não-estruturado, visto que há diversas formas de representar, em uma frase, a mesma informação e significado. Informações valiosas podem ser extraídas desses textos, como opiniões a respeito de produtos, serviços e até mesmo opiniões sobre pessoas. Empresas e provedores de serviços podem usar tal conhecimento para melhor entender como os seus consumidores os veem e se estão ou não satisfeitos com seus produtos e serviços. Com a enorme quantidade de documentos disponível, categorizá-los manualmente é uma tarefa inviável. Para esse fim, a Análise de Sentimento (AS), uma subárea do Processamento de Linguagem Natural que almeja automaticamente inferir a polaridade ou orientação do sentimento expressado (positivo, negativo ou neutro) em um documento pode ser empregada.

Através da categorização de documentos, de acordo com o sentimento específico expressado neles (felicidade, tristeza, etc) ou orientação de polaridade (positiva, negativa, neutra), pode-se obter uma melhor e menos laboriosa análise deles, pois diminui a quantidade de trabalho manual exigida. Faz-se importante entender as ferramentas e suas limitações a fim de construir soluções com melhor desempenho em termos de acurácia e mais genéricas, podendo ser aplicadas em diferentes domínios.

Para realizar o aprendizado supervisionado de documentos, tarefa esta também chamada de classificação, características (*features*) devem ser extraídas dos documentos não-estruturados, de forma que elas possam ser utilizadas como entrada em algoritmos de classificação. Este trabalho apresenta um estudo e análise comparativa de representações densas e esparsas de documentos, técnicas de pré-processamento de texto, e algoritmos de aprendizado aplicados à tarefa de classificação binária de polaridade de sentimento (positiva ou negativa) em duas configurações experimentais envolvendo avaliações no mesmo domínio e em diferentes domínios.

As comparações entre tais configurações experimentais são realizadas de acordo com a macro-média da métrica F1 (F1-Score), que leva em consideração precisão e cobertura. Um total de cinco *datasets* de dois domínios distintos foi utilizado, sendo três deles do domínio de avaliações de filmes e os outros dois do domínio de avaliações de produtos de diversas categorias.

Palavras-chave: análise de sentimento. mineração de opinião. aprendizagem de máquina.

Abstract

In recent years, there has been an exponential growth in publicly available content on the Web. A relevant amount of those documents are presented in the form of Natural Language Text, which is inherently unstructured, as there can be many forms to express the same information and meaning. Valuable information can be extracted from those contents, notably people's opinions on products, services and even about other people. Companies and service providers can use that knowledge to better understand how their customers think about them or to know how their products and services are appreciate of not by their customers. Given the sheer amount of available information, manually categorizing such documents becomes an unfeasible task. To that end, Sentiment Analysis, a subfield of Natural Language Processing which aims to automatically infer the sentiment expressed in an unstructured natural language document, can be employed.

Categorizing documents according to their specific sentiment (happiness, sadness, etc) or polarity orientation (positive, negative, neutral) enables better and less laborious analysis of them by decreasing the manual effort required. It is also important to understand the available tools and their limitations to develop more robust and efficient solutions.

In order to perform supervised learning for document classification, descriptive features must be extracted from the unstructured documents, in such a way that they can be fed into classification algorithms. This work presents a study and analysis of sparse and dense document representations, text preprocessing, and learning algorithms in two experimental settings involving evaluation on the same domain and cross-domain concerning the task of binary sentiment polarity classification (positive or negative). The comparisons are based on the macro-average F1-Score, which takes into account both precision and recall. A total of five datasets from two distinct domains were used, being three from movie reviews and two from customer's reviews of products from multiple categories.

Keywords: sentiment analysis. opinion mining. machine learning.

List of Figures

List of Tables

List of abbreviations and acronyms

AZ	Amazon Dataset
BOW	Bag-of-Words
CR	Customer Reviews Dataset
IMDB	Internet Movie Database
LR	Logistic Regression
NB	Naïve Bayes
NLP	Natural Language Processing
RF	Random Forest
RT	Rotten-Tomatoes
SA	Sentiment Analysis
SPC	Sentiment Polarity Classification
SST	Stanford Sentiment Treebank
SVM	Support Vector Machine
WE	Word Embedding

Contents

1 Introduction

In recent years, there has been an exponential growth in publicly available content on the Web. A relevant amount of those documents are presented in the form of natural language texts, which are inherently unstructured. Moreover, the Internet experienced a transition where users stopped being only content consumer and became content creators, advent popularized by the name of Web 2.0 (CISCO, 2015).

Valuable information can be extracted from unstructured, natural language texts, notably people's opinions on products, services and even about other people. This kind of information can be very useful, for instance, for companies and service providers that can use this knowledge to better understand how their customers evaluate them or to know how their products and services are appreciated by their customers.

For processing such unstructured texts, Natural Language Processing (NLP), a subfield of Computational Linguistics, which focus on the automatic comprehension of human language can be used for enabling computers to extract valuable information (RAVI; RAVI, 2015).

In this context, Sentiment Analysis (SA), a subfield of NLP which aims to automatically infer the sentiment expressed in natural language documents, can be employed for extracting such opinions (LIU, 2012) (PANG; LEE; VAITHYANATHAN, 2002) (TURNEY, 2002). The main goal of SA is to analyze opinions, sentiments, attitudes, and emotions from people about products, companies, services, topics, and even other people (LIU, 2012). The SA field and related tasks are relatively new, dating from two decades ago, with the first studies being conducted by (WIEBE, 2000) (PANG; LEE; VAITHYANATHAN, 2002) (TURNEY, 2002).

SA has been largely explored by companies and businesses interested in monitoring what their customers think about them, their services and competitors, as well as for better understanding their interests. It has also, along with other NLP tasks, been used in electoral campaigns, such as both the 2008 and 2012 Barack Obama campaigns in the United States, which had a specialized data mining team analyzing its success from people's opinions on social media (CLASTER, 2016). In addition, increasingly more companies have started basing their business decisions on social media. This created a need for specialized SA service-providers tailored to other businesses, in order to better understand their position on the market (LIU, 2012).

Another major source of content extensively explored in SA is social networks such as Twitter, with approximately 500 million tweets (posts) every day (TWITTER, 2013),

containing a large amount of opinions on various subjects. Such kind of social media data poses new challenges due to the the number of character limit and informal environment, leading to many word abbreviations and difficult text preprocessing (RAVI; RAVI, 2015).

SA has enabled several useful applications according to (RAVI; RAVI, 2015): **Stock market prediction**, such as the work described by (BOLLEN; MAO; ZENG, 2011), where sentiment information from Twitter posts is used to forecast the market. **Box-office prediction**, in which people's opinions about the initial release of a movie can be used to predict how well the movie will do, for example. **User satisfaction assessment**, where information from social media sites such as Facebook and Twitter, or a company's own feedback center can be used to measure the overall response of users towards their company or services, for example. **Targeted advertising**, making use of information opinions from users to model their preferences and produce ads with a larger rate of conversion, by captivating the user's interest. (RAMBOCAS; GAMA et al., 2013) describe the importance of SA to marketing.

Studies on approaches to SA have been previously conducted. In (LIU, 2012) and (MEDHAT; HASSAN; KORASHY, 2014), it is provided an overview of tasks, algorithms, approaches, obstacles, and applications. (SERRANO-GUERRERO et al., 2015) compares free (and limited free use) web-services for SA and discusses their advantages. (RAVI; RAVI, 2015) surveys, at a finer-grained level, the different approaches in the literature.

This work focuses on the task of Sentiment Polarity Classification (SPC) which categorizes documents according to their polarity orientation (positive, negative, neutral). Many studies has already been proposed to the SPC problem in which the most successful ones are based on supervised machine learning (KHARDE; SONAWANE et al., 2016), i.e., the task of learning a solution to a problem based on labeled instances.

In order to classify the polarity of reviews and people's opinions on texts, they first have to be converted to some king of representation as input to supervised machine learning algorithms.

One of the widely used representation is the so-called Bag-of-Words (BOW) representation that maps variable length texts into a fixed dimensional vector, represented by a finite vocabulary. In other words, BOW representation treats a document as an unordered set of word, and uses the frequency distribution of them as the primary evidence in the classification (BESPALOV et al., 2012). BOW has the advantage that as the vectors are highly sparse, they are computationally easy to represent. However, it fails to capture syntactic and semantic similarities that are easily discoverable in data, e.g., "pretty" and "beautiful" have similar meanings; opposite to "unattractive" and "ugly".

More recently, distributed word representations, a.k.a. Word Embeddings (WE), have been proposed and they showed to benefit several NLP tasks, including parsing, and

sentiment analysis (SOCHER et al., 2011). WE are created by applying neural networks to obtain dense, low-dimensional, continuous word representations (BENGIO et al., 2003). WE main advantages resides in the fact that they not only can be derived directly from raw, unannotated corpora, but also can capture several aspects concerning lexical semantics (TURNEY; PANTEL, 2010).

The BOW features, which are count-based, are generally very sparse, because many words do not occur in all the documents. For that reason, they are also called sparse representations. On the other hand, WE features are lower-dimensional real-valued vectors and are also known as dense representations.

There is a large number of other factors to be considered when performing SA. They vary the from level of granularity to the choice of feature extraction and learning algorithms employed. Many of the recent works comparing such factors are limited in the number of aspects evaluated (KHARDE; SONAWANE et al., 2016). Furthermore, the literature review conducted by this work showed that a deeper exploration of the results derived from an fair assessment in a cross-domain setting is still largely unexplored in SA. Finally, the findings derived from this work might influence further research and the deployment of SA and other text classification techniques to more realistic SA application scenarios.

Research Questions. The proposed study and experiments described in the following chapters aim to answer and provide cues for the following research questions:

1. Which features are better suited for polarity classification in SA?
2. Which learning algorithms have better performance for polarity classification in SA?
3. How does preprocessing affect polarity classification?
4. Is it possible to obtain effective polarity classification models when the training dataset is different from the testing one? (same domain or a different domain)? And how this can affect the overall performance?

1.1 Research goals

The main research goal of this study is to provide, quantitatively and qualitatively, analyses of sparse and dense features, learning algorithms, and preprocessing techniques on datasets from various domains at a document level, under the same evaluation settings across all the experiments.

The main goal can be decomposed in the following specific goals:

1. investigating the best sparse and dense features for the task of binary sentiment polarity classification on the selected benchmark datasets.

2. comparing the performance of different supervised learning algorithms on various datasets.

3. evaluating the generalization level of the classification models generated by the supervised machine learning algorithms when using two different datasets: one for training and the other for testing (same domain and cross-domain evaluation scenarios).

4. presenting analyses and discussions on the results obtained from the experiments, as well as highlighting the major influencing factors.

1.2 Document Structure

The remainder of this document is structured as follows: Chapter 2 introduces and defines the techniques involved in the execution of this work. Chapter 3 presents an overview of the literature on the field, main works and their contributions, as well as limitations and research gaps; Chapter 4 details evaluation methodology for investigating the research questions proposed in this section. Chapter 5 addresses the details of the execution of the study and presents analyses of the results obtained from the experiments. Lastly, Chapter 6 describes the limitations, possible improvements and continuation to this work in the future.

2 Background

This chapter introduces the main concepts and techniques employed in this work.

2.1 Feature Generation

In order to perform Sentiment Analysis, a set of describing attributes (features) has to be extracted from the text. In the following, the features used for representing the reviews in this work are briefly presented.

2.1.1 Count-Based

Count-based are features based on the number of occurrence of words in the document. The vocabulary can be very big, and most words do not occur in every document, but rather in a portion of them. This leads to sparse vector representations of a document, which is the reason why such features are also known as sparse document representations.

2.1.1.1 N-grams

N-grams are groups of contiguous words. They can be formed by just one word to as many as the total number of words in the document. This size, naturally, impacts the descriptive information which can be obtained. For example, in the sentence "the sky is blue", the 2-grams (or Bigrams) are "the sky", "sky is" and "is blue". The occurrence count of those bigrams (or any-sized n-grams) can be used as input to some machine learning algorithm. Making this number too small can make the generated feature unable to catch particular details from the text (negations, for example) and having it too large can lead to very rare n-grams which are not present in most of the future documents to be analyzed. In this work, features are generated from 1-grams (unigrams), 2-grams (bigrams) and 1-3-grams (ngrams), i.e, 1, 2 and 3-grams.

2.1.1.2 TF-IDF

Term frequency times inverse document frequency is a feature that consists of a weighted histogram of word frequencies. The terms are weighted by its inverse document frequency (JONES, 1972), i.e., in what percentage of documents the term appears. This kind of weighting allows certain terms to have stronger influence when they are more specific to a few classes and weaker when they appear too often in diverse kinds of

documents. The Equation 2.1 shows how the weighting term is calculated, where $df(d,t)$ is the number of all the documents containing the term t.

$$idf(t) = log\frac{1+n_d}{1+df(d,t)} + 1 \tag{2.1}$$

This feature is essentially the same as the N-grams, but additionally weighted by the *idf* term described in Equation 2.1, and normalized by the Euclidean norm, as shown in Equation 2.2, meaning the resulting vectors have magnitudes between 0 and 1 in each dimension.

$$v_{norm} = \frac{v}{||v||_2} = \frac{v}{\sqrt{{v_1}^2 + {v_2}^2 + \cdots + {v_n}^2}} \tag{2.2}$$

2.1.2 Word Embeddings

Word Embeddings are dense representations of words. That is, smaller dimensional real-valued vectors, rather than one-hot. A one-hot representation of a word is a 1-by-N vector (where N is the vocabulary size) with one of the dimensions set to one and the remaining set to zero.

This technique of dense representations is not entirely new (BAKER; MCCALLUM, 1998), but recently many derived techniques to learn those embeddings faster have gotten attention, such as Word2Vec and GloVe. The main advantage of dense vectors compared to sparse vectors is that words with close meaning also have closer representations, e.g., cat and dog can be used interchangeably in many situations and this causes them to have similar representations.

2.1.2.1 Word2Vec

Word2Vec (MIKOLOV, 2013) is a very recent technique for learning word embeddings. It learns, in an unsupervised manner, representations of words as vectors, hence the name. These vectors also have particularly useful relationships in their space, for example, "King" - "Man" + "Woman" = "Queen". This means that subtracting the vector for the word man from the word king and adding the vector of the word woman is somewhat close to the vector representation of queen. Word2Vec embeddings are generated using a Neural network-based language model similar to (BENGIO et al., 2003). For example, given the context (surrounding words), predict the word. Figure 1 demonstrates a few word relations that can be found in the Word2Vec embedding.

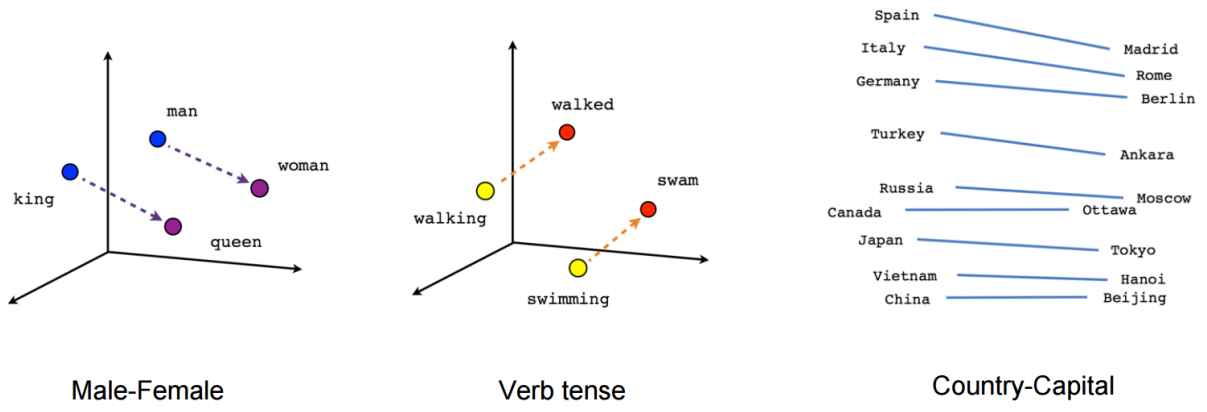

Figure 1 – Examples of linear relations between words that can be found in the Word2Vec embedding.

Source: (TENSORFLOW. . . , 2016)

2.1.2.2 GloVe

Global Word Vectors (PENNINGTON; SOCHER; MANNING, 2014), or GloVe for short, are learned differently from Word2Vec embeddings. Rather than using neural networks in a predictive fashion, the co-occurrence matrix of words and contexts are used and then factorized into a lower dimensional matrix, from which representations for each word in the vocabulary are extracted.

2.1.2.3 Dependency-Based Word-Vectors

Extension of the Word2Vec algorithm, but instead of using the context words in a window centered in the word under analysis, the context words in the Dependency Parse Tree is used. This allows to capture relation between words that are further apart than the proximity window considered when using the regular Word2Vec algorithm. (LEVY; GOLDBERG, 2014)

2.1.2.4 AffectiveSpace Embedding

The AffectiveSpace embedding is generated from concepts extracted from text. Differently from the other embeddings used in this work, which are embeddings for words. Concepts are, as described in (PORIA et al., 2014), groups of words with specific meanings that wouldn't be captured by using individual words.

2.1.2.5 ConceptNet Numberbatch

Numberbatch (SPEER; CHIN; HAVASI, 2017) is an ensemble of word embeddings created by using data from ConceptNet (a knowledge graph), Word2vec and GloVe embeddings, and OpenSubtitles dataset of subtitles for movies and TV shows. The original

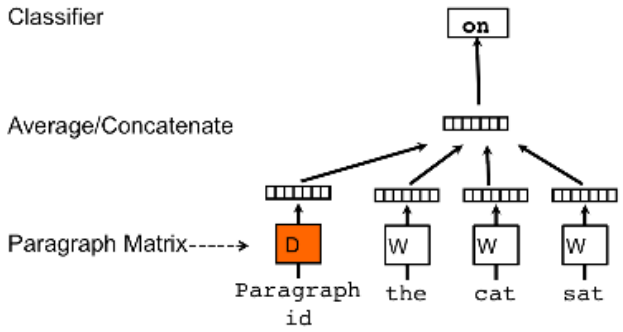

Figure 2 – Doc2Vec embedding generation scheme.

Source: (LE; MIKOLOV, 2014).

word embeddings from GloVe and Word2Vec are used as a starting point and are fine-tuned using additional data.

2.1.2.6 Doc2Vec

Doc2Vec, or Paragraph2vec (LE; MIKOLOV, 2014), is an extension to the Word2Vec embedding generation algorithm which works utilizing labeled sentences. This extension allows a new embedding to be trained taking the class labels into account.

Figure 2 shows the Neural Network setting where embeddings are learned. The only difference from the original Word2Vec scheme is the aditional Paragraph ID, which is a token added to the document. Embeddings for this new token are also learned and they represent the document vector.

2.2 Learning Algorithms

This section briefly describes the supervised learning algorithms addressed in this study. Machine Learning algorithms try to model automatically model a solution or representation of specific problem by observing instances. Supervised algorithms, in particular, learn from labeled instances, i.e., instances for which the class is known - for classification problems. By learning a mapping from a sample's characteristics to its class, supervised algorithms aim at producing a general function which also correctly classifies unseen samples during the training.

2.2.1 Bernoulli Naïve Bayes

Naïve Bayes is a probabilistic classifier based on the Bayes' Theorem. It assumes the probabilities are unrelated/independent which is rarely ever true, hence the name naïve. Despite this, it works reasonably well and is a very standard algorithm for Sentiment

Analysis. The Bayes' Theorem describes the probability of an event, given some prior information. It is shown in Equation 2.3, where $P(A|B)$ is the conditional probability of an event A, e.g. the document class being A, given an observation B, and $P(B|A)$ is the probability of observing B given that A is true.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{2.3}$$

The Bernoulli variant of the NB classifier assumes features to be binary, indicating the presence or absence of a feature which are words in the case of text classification. This variant is characterized by how feature distributions are estimated. The Bernoulli estimates the probability of a term given a class $P(i|y)$ to be the fraction of documents in class y where the term i is present (MANNING; RAGHAVAN; SCHÜTZE, 2008). Its decision rule is based on Equation 2.4, and the class is predicted to be the one with higher score, i.e., Equation 2.5.

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i) \tag{2.4}$$

$$\hat{y} = argmax P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.5}$$

2.2.2 Logistic Regression

The Logistic Regression or Maximum Entropy classifier is a statistical way of modeling the best fitting solution to the relationships of involved descriptive variables (the features extracted from the corpus, in this case). It was first introduced by (COX, 1958).

The LR model attempts to learn a transformation of the input features, while minimizing a cost function. It is closely related to the Multilayer Perceptron (MLP) model, which also works by learning the parameters of a logistic-squashed transformation. The Equation 2.6 presents the LR inference function, where θ are the model's parameters, except it is optimized by the Coordinate Descent algorithm, while MLP's parameters are learned by Backpropagation and Gradient descent.

$$F(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{2.6}$$

2.2.3 Support Vector Machines

Support Vector Machines are a family of classifiers that find, deterministically, the best dividing point between classes. That is, a hyperplane situated between the boundary regions of each class that is equally distant from each one of them. The SVM learning algorithm was first described in (BOSER; GUYON; VAPNIK, 1992).

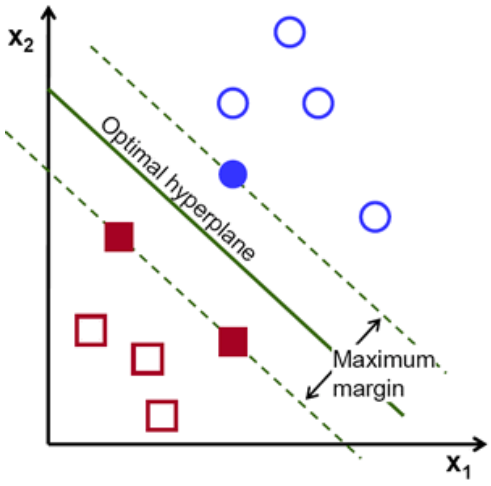

Figure 3 – Optimal hyperplane with maximum distance between class boundaries on a SVM.

Source: OpenCV online documentation

The SVM training algorithm works by finding the boundary regions between the classes, and using them to find the best separating hyperplane. The training samples used to find this hyperplane are called support vectors. Figure 3 depicts a hyperplane which satisfies the maximum-margin property. The filled squares and circles are the support vectors, which are the only instances that need to be kept to make a model. The hollow squares and circles are other samples. New samples are classified according to which side of the optimal hyperplane they lie in.

2.2.4 Random Forests

Random Forests are a kind of ensemble classifiers, i.e., classifiers which are based on groups of predictors, that are made of decisions trees generated by training on different chunks of the training data. This chunking of the data (also known as bootstrap aggregating or bagging) confers Random Forests more robustness to overfitting by smoothing the variance observed in the sampled training data.

Each of the classifiers in the ensemble are Classification and regression trees (CART) decision tree (BREIMAN et al., 1984). CART decision trees are constructed by iteratively finding the feature which best separates the classes and specializing further down the tree. Instead of a separating line (or hyperplane), the decision boundary of a single decision tree can better be explained by a set of lines parallel to the axes.

3 Literature Review

This chapter aims at briefly reviewing the literature on Sentiment Analysis. Several works, in which this study is based upon, are discussed along with their main contributions to the development of the field.

Sentiment Analysis (SA), as a subtask in Text Mining, aims at inferring a specific sentiment (happiness, sadness, etc) or polarity (positive, negative, neutral) from natural language texts.

The following subsections present SA concerning the level of granularity and techniques involved.

3.1 Granularity

The granularity in SA is the scale at which documents are taken into consideration. For example, one can consider the document as a whole and summarize its overall orientation towards being positive or negative; or determine the orientation of specific aspects or features of the entities involved. Another distinct aspect concerns whether the opinions are explicitly or implicitly expressed in the text. As an example of the later, consider the sentence "The ending is not surprising", in which a person implicitly talks negatively about the plot of the movie, even though it never mentions "plot".

The main granularity levels are:

Lexical: where words are considered individually, regardless of where or how they are found in a specific phrase, in order to determine the sentiment. For example, determining the polarity of a specific word.

Syntagmatic: groups of words are considered, i.e., which words precede or follow other words. In (MATSUOKA; LEPAGE, 2014), this is referred to as co-occurrences and they are utilized as evidence to extract people's opinions from phrases.

Sentence: sentences from a document are taken into account separately, identifying for each of them if the sentiment expressed is positive, negative or neutral, or to identify the presence of lack of opinative information (subjectivity detection), as in (WIEBE; BRUCE; O'HARA, 1999).

Document: the whole text is taken into account in order to classify the sentiment. This is closely related to sentence level, as sentences can be seen as short documents.

Aspect: specific features of an entity are considered. For example, one might want to automatically infer from the text what are the customer's thoughts on look, weight and

performance of a smartphone.

In this work only document-level techniques are considered.

3.2 Learning Technique

In addition to level of granularity, the learning technique is also another way SA approaches can be grouped by. The learning technique comprises of the methods used to perform the training, if applicable, and inference of the model. Most SA approaches are performed using hand-engineered rules or features, Lexicons or Machine Learning.

The Figure 4 presents an overview of common SA approaches, which are discussed in the following paragraphs.

Rule-based approaches are constructed based on human-observed patterns from the corpus that are used to perform the sentiment classification. Rule-based systems are a kind of Expert System, which are systems specifically designed to mimic a highly-capable human's decision-making (JACKSON, 1986).

Machine Learning approaches use either hand-engineered or statistical features extracted from the corpus as input to learning algorithms, such as SVM, Logistic Regression or Naïve Bayes. ML approaches can be either supervised or unsupervised. Supervised methods rely on learning from labeled samples, while unsupervised methods do not require labeled training data, such as in (ROTHFELS, 2010) where "seed" polarity words are used to further extract other polarity terms and using a sentence scoring function based on these terms. While unsupervised methods are interesting due to not needing to label any data, supervised methods generally perform better.

Lexicon-based approaches make use of word polarities and word relations (synonyms, antonyms, group of concepts) present in Lexicons, which serve as a sort of "sentiment dictionary" (MILLER, 1995). An example of simple Lexicon-based and rule-based approach would be to determine the document polarity by looking at the polarity of words that compose it. However, sentences such as "This dish is not very good" could come out as neutral or even positive when it actually expresses a negative sentiment. It is subdivided in two methods: dictionary-based and corpus-based. The dictionary-based method uses only information from the Lexicon, where synonyms, antonyms and other related words' polarities are also considered; and the corpus-based approach where additional data from large corpora are used to aid the finding of other opinion words, with additional context information to their polarities.

There are also hybrid approaches, which combine multiple of the aforementioned, though they are less common due to their higher computational cost (MEDHAT; HASSAN; KORASHY, 2014).

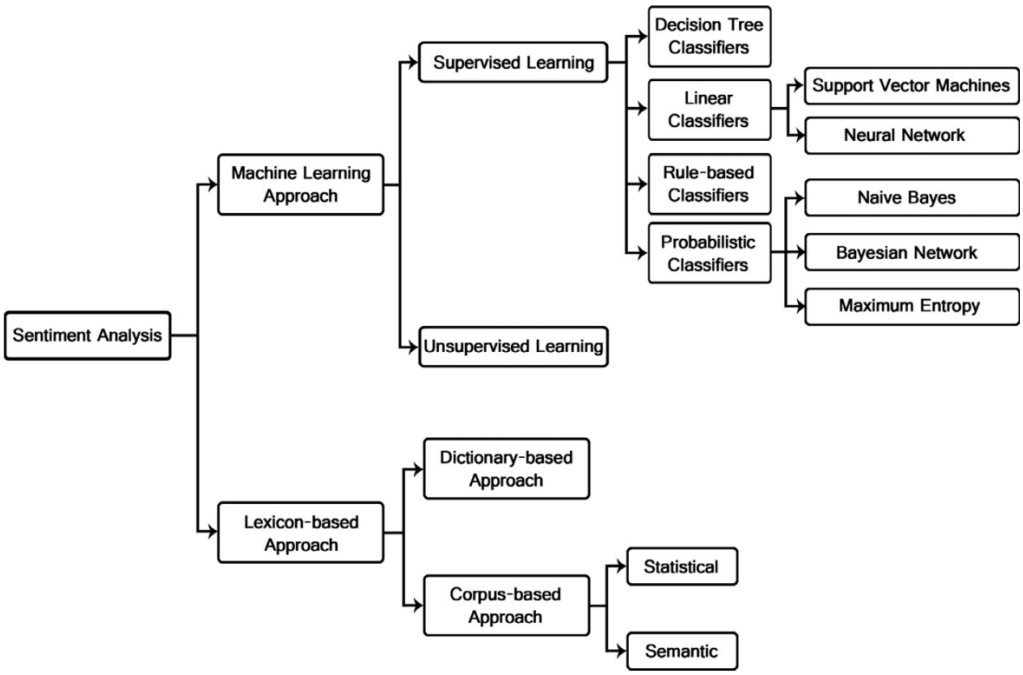

Figure 4 – Overview of Sentiment Analysis approaches.

Source: (MEDHAT; HASSAN; KORASHY, 2014)

In general, rule-based and dictionary-based approaches, such as the ones used in the system proposed by (QIU et al., 2010), are phasing out and losing popularity to purely ML-based techniques.

3.3 Related Work

In this section, some works in the field of Sentiment Analysis and NLP are presented, along with their main contributions and unexplored gaps which this study intends to bridge.

In (TURNEY, 2002), one of the earliest works in the field of Sentiment Analysis, an unsupervised technique for sentiment classification based on Point-wise Mutual Information is proposed, where the PMI between the phrase and specific words chosen to represent positive and negative meaning is compared and used to determine the semantic orientation of the phrase.

In (LEOPOLD; KINDERMANN, 2002), different frequency-based features for text presentation are evaluated using Support Vector Machines for the problem of text classification. They also show experimentally that the choice of feature is more crucial than the choice of kernel function for the SVM.

(PANG; LEE, 2004) proposes a subjectivity classification step before performing sentiment classification. This allows objective sentences, which don't have any sentiment

associated, but are rather factual, to be filtered out. After this step, sentiment polarity classification is performed using unigram-presence feature with both NB and SVM classifiers, evaluated on a movie review dataset extracted from Rotten Tomatoes.

(MANNING; RAGHAVAN; SCHÜTZE, 2008) describes in details how the Naïve Bayes algorithm can be used to perform text classifications, and presents some foundational knowledge on some of its properties.

A method for dealing with negation in sentences is examined in (WEI; GULLA; FU, 2010), where negating expressions are counted and used to flip, when applicable, a sentences predicted polarity. They also propose an algorithm for generating multi-unigram features, which are described as frequent-occurring term combinations. Although showing some prominent results, they also report that the proposed technique falls short when the negation is subtly presented, i.e., hidden in the sentence.

The impact of using preprocessing techniques in the task of Sentiment Analysis and Text classification is analyzed in (CAMACHO-COLLADOS; PILEHVAR, 2017) in the specific context of Neural Networks. They analyze lemmatizing, lowercasing and word-grouping for various SPC and Text Categorization datasets. Their findings show that many these techniques generally yield worse results, except for very specialized domains.

In (MEDHAT; HASSAN; KORASHY, 2014) and (RAVI; RAVI, 2015), algorithms and applications on the literature are compiled and categorized in a survey. They also include information about what techniques and datasets were used in each work.

In (WU; TAN, 2011), a technique for cross-domain sentiment classification is proposed and evaluated. The work proposes a two-step framework for bridging knowledge from one domain to another. Other challenges and solutions of cross-domain SA are also covered by (KURIAN, 2014).

(Rushdi Saleh et al., 2011) explore SVM utilizing various n-gram features and weighting schemes for evaluation on datasets of different domains in order to access how sentiment classification is affected. However, they don't evaluate how models perform when tested on another dataset (different from the training dataset).

The problem of unbalanced datasets is discussed in the study carried in (BURNS et al., 2011). They report that unbalanced datasets consistently perform better with more data, while for balanced datasets that may not always be the case.

An end-to-end SA solution is proposed by (PORIA et al., 2014) in a hybrid approach using dependency-based rules, concept extraction and Lexicon of word polarities in a 4-dimensional spectrum (pleasantness, sensitivity, aptitude, attention).

Deep Learning methods for Sentiment Analysis are discussed in (TANG; QIN; LIU, 2015) and (ROJAS-BARAHONA, 2016). The former also includes analysis for other tasks

related to SA such as Lexicon construction, while the latter evaluates different architectural type (sequential, recursive and hybrid) for polarity classification (binary and fine-grained).

A study of evaluation methods for Word Embedding is shown in (SCHNABEL et al., 2015). They show how embeddings trained on the same data and with the same objective function can encode different information. Baselines and results utilizing embeddings are also shown for different text classification tasks, including Sentiment Analysis.

Overall, the related works discussed in this section go over different feature generation techniques, preprocessing, algorithms or some form of preliminary steps to aid classification by, e.g., filtering out irrelevant information or dealing with unbalanced data. To the best of our knowledge, no other work has focused on cross-dataset evaluation of multiples features, word embeddings and learning algorithms, as well as preprocessing impact on classification.

4 Methodology

This chapter presents details of the adopted methodology for evaluation of the techniques described in Chapter 2 for the task of binary polarity classification in Sentiment Analysis.

In order to guarantee flexibility and reproducibility when evaluating different feature extraction methods and learning algorithms, all the experiments followed a solid pipeline. The pipeline ensures all algorithms receive the same inputs under the same circumstances, and is composed by modules with specific purposes, which are described in the following sections.

Figure 5 shows an overview of the complete pipeline and how the modules are connected.

In the following, each component of Figure 5 is described in detail.

4.1 Entry-point

The entry point consists of a main script which processes user input (the experiment settings), and interconnects the other components in the pipeline. This component handles the execution of experiments and its configurable options are:

- **Datasets:** a list of datasets included in the experiment. All the datasets available

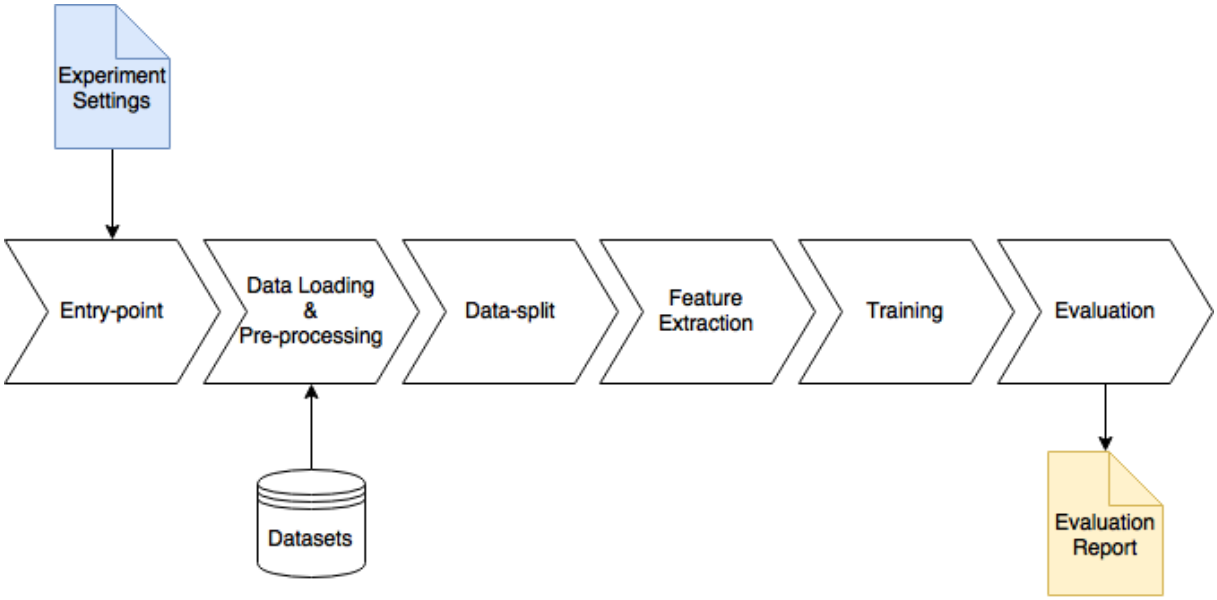

Figure 5 – Evaluation pipeline.

Source: The author

Figure 6 – Data preparation pipeline.

Source: The author

to the system must be already preprocessed (cleaned) and formatted as comma-separated files (.csv). The datasets used in this study are introduced and discussed in Section 5.1.

- **Features:** list of feature types which are going to be extracted from the documents in the provided input datasets. This study includes several feature extraction algorithms (see Section 2.1) and can be extended by adding functions that implement the same signature.

- **Classifiers:** a list of classifiers which are going to be trained with the aforementioned features. The classifiers present in this study are discussed in Section 2.2.

4.2 Data Loading

This component is responsible for handling loading operations of the datasets. The datasets to be loaded are specified in the experiment setting options provided by user input. The selected datasets are from different sources and are available in different formats (XML, plain-text, etc). For that reason, it was first necessary to define a common structure and standardize their formats. The chosen structure was the comma-separated file, which is a plain-text file format, where table-like structures are saved a row per line and each column-value separated by a comma.

This first step required manually designing scripts to handle each dataset structure. Then, after all the datasets standardized in the same format, the proposed system is able to load them with no additional step required. Some datasets were directly scraped off from the Internet and had documents containing HTML tags, unrecognized characters (different encoding) or missing core information (the user review itself).

These scripts were also responsible for handling the above-mentioned problems before generating the final .csv file. In Figure 6, there's an illustration of the data preparation pipeline.

The csv file structure contain the following fields:

- **Document:** the plain-text document sample from the dataset.

- **Polarity:** the sentiment orientation of the sample (positive or negative)
- **Split:** optional, which subset of the data the sample belongs to. Since some datasets have predefined train, validation and test splits, this information is kept at this point.

The preprocessing part of this component removes extra whitespaces and non-alphanumerical characters except for question marks, exclamation marks, parentheses, commas, and quotation signs, as such characters are often used to emphasize sentiment in text. Lowercasing and stop-word removal are not taken into account in this step in this study and are, instead, configurable options in the count-based feature transforms, as described in Section 4.4.

4.3 Data Splitting

This component handles the splitting of each input dataset. It operates in two ways, depending on whether the evaluation setting is single-dataset or multidataset. They work as follows:

- **Single:** in the case where a single dataset is considered, the dataset is split into 10 folds, from which the cross-validation is performed. In other words, it trains K different models using different chunk as testing data at each time, while the remaining K-1 data splits are used as training data. This procedure is known as K-fold cross-validation and is discussed in (JONES, 1987).
- **Multidataset:** in the case where multiple datasets are used as input, each one of the datasets is used as either training or testing data in a given experiment. No further splitting of the individual datasets is done.

All of the splits in the single-dataset setting are performed in a stratified manner. That is, the class balance distribution is kept across each split of the original data. This is performed in order to avoid evaluation on test data that is too different from training data.

4.4 Feature Extraction

This is the module where all the feature extracting functions are implemented. All of these feature extractors share a common interface which takes the data chunks as input and generates the features for each of them.

Model	Source
Word2Vec	Google Code Word2Vec repository [a]
GloVe	Stanford GloVe project page [b]
AffectiveSpace	SenticNet website [c]
Dependency-based	Omer Levy's blog [d]
Numberbatch en17.04b	Conceptnet Numberbatch project page [e]

Table 1 – Pre-trained Word-Embedding models' sources.

[a] https://code.google.com/archive/p/word2vec/
[b] https://nlp.stanford.edu/projects/glove/
[c] http://sentic.net/affectivespace.zip
[d] https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/
[e] https://github.com/commonsense/conceptnet-numberbatch

In the case of the word-embedding features (with the exception of Doc2Vec, which involves a training step), the extraction process is actually a simple look-up operation, retrieving the dense representation of each word for the word embedding defined by the user, and finally averaging all the vectors to form the vector representing the whole document. The pre-trained embedding models were obtained from various sources, as shown in Table 1.

For the remaining features, which are transformations dependent on word occurrence distributions in texts, the testing data must not be present in the training stage in order to not include testing information; that's the reason this module comes after the data splitting module. Such features also include a few configurable parameters, including the additional preprocessing steps stop-word removal and lowercasing. Therefore, two versions of each of these transformation features are employed in this study: with and without the additional preprocessing.

4.5 Training

In this step of the pipeline, the selected learning algorithms are trained using the input datasets. In short, after the features corresponding to each subset of data have been extracted, the system uses them as input to each learning algorithm, that generates the corresponding classification models.

The predictions on the test data are generated at this stage and they are passed to the subsequent model, where the assessment process is executed.

4.6 Evaluation

The evaluation module takes the predictions on the testing data and generates the metrics for each trained model. The metrics are also persisted in .csv files. The structure of the evaluation report depends on the user input options concerning whether the training process involves a single model or cross-validation. For the multimodel evaluation setting, i.e., where multiple datasets are entirely used as training and testing data, the evaluation reports the evaluation metrics for each classifier. For the case in which multiple models are trained, i.e., where single datasets are used, the metrics for each model, as well as mean and standard deviations across all the folds, are included.

5 Experimental Setup

This section describes the experimental setup (datasets, evaluation metrics, machine learning algorithms and their parameters) that were used to perform the experimental evaluation for answering the research questions mentioned in the introductory part of this document.

5.1 Datasets

This section introduces the datasets used in the experiments and important details regarding their structure. A total of 5 datasets was used: three from the movie reviews domain and two from product reviews domain. Table 2 lists the datasets and their characteristics regarding average document length and class distribution.

Dataset	Document Length	Positive	Negative	Total
Rotten Tomatoes (rt)	23.35	5331	5331	10662
Stanford Sentiment Treebank (sst-2)	19.97	4963	4650	9613
IMDb (imdb)	269.28	25000	25000	50000
Amazon Reviews (az)	214.93	17422	4932	22354
Customer Reviews (cr)	20.59	2405	1366	3771

Table 2 – Datasets and statistics about average document length (tokens) and class distribution.

The Rotten-Tomatoes dataset consists of movie reviews extracted from the Rotten-Tomatoes website. It was introduced in (PANG; LEE, 2005) and contains short reviews equally balanced across positive and negative. This dataset is one of the most widely used datasets in the literature.

The Stanford Sentiment Treebank (SOCHER; PERELYGIN; WU, 2013) is derived from the Rotten-Tomatoes (RT) dataset. It was parsed at a more granular level (phrases, instead of whole documents) and annotated according to their sentiment. This dataset was also annotated using a fine-grained sentiment classification scheme including very negative, negative, neutral, positive and very positive. However, since this work focuses on binary sentiment polarity classification, the neutral class is ignored, while the negative and very negative classes are merged, and similarly for positive and very positive. In the literature, the fine-grained annotation of this dataset is referred to as sst-1, while the binary annotation is referred to as sst-2.

The IMDb dataset, similarly to the Rotten Tomatoes dataset, was scraped from the Internet Movie Database, which - alongside movie and cast information - also includes

a user review section. It was introduced in (MAAS et al., 2011) and is, together with the Rotten Tomatoes dataset, one of the most widely used datasets in the literature for Sentiment Analysis, as well as some other NLP-related tasks. In addition, this dataset contains more documents than the others.

The Amazon Reviews dataset also features user reviews extracted from product pages in Amazon. It has a balanced amount of positive and negative reviews and was originally introduced by (JO, 2011).

The Customer Reviews dataset includes user reviews for 5 different products extracted from Amazon, including digital cameras, cellular phones, MP3 players, and DVD players. This dataset was first introduced by (HU; LIU, 2004).

The RT, SST-2 and IMDB datasets are from the movie reviews domain, while the AZ and CR are based on customer's product reviews.

5.2 Feature Extraction

The features used for the evaluation are, as described in the Chapter 2, distributed in two groups of features, namely count-based features and word embedding features. Count-based features are derived after some transformation applied to the documents and they are based on the vocabulary of the corpus. Word embedding features are not necessarily learned from the same corpus and can be - and usually are - learned in an unsupervised fashion.

For the count-based features, Unigram, Bigram, N-gram and TF-IDF weighted Ngram are considered, including a further version of each of them where an additional preprocessing step was performed. The preprocessing step removes English stopwords from the document and lowercases all the words, i.e., there's no difference between capitalized and uncapitalized versions of the same word. For all datasets, the maximum number of generated features are limited to a maximum of 5000, meaning that no more words are going to be considered in the vocabulary; and the most common ones being prioritized.

For the word embedding features, the following embeddings are considered: Word2Vec, GloVe, Dependency-based, Numberbatch, and AffectiveSpace.

5.3 Learning Algorithms and Hyperparameter Settings

The learning algorithms addressed in this study are the aforementioned Naïve Bayes, Support Vector Machine, Logistic Regression and Random Forest. Hyperparameters are parameters unrelated to the problem, i.e., SA in this study, that control how the algorithms learn a model. This section presents more details about classifiers' hyperparameters and

implementation. Hyperparameter optimization was not performed as it is not in the scope of this study. All of the algorithm implementations used the implementations provided by the scikit-learn[1] library (PEDREGOSA et al., 2011), a general Machine Learning library for the Python[2] programming language.

The Naïve Bayes is based on the Bayes Theorem and assumes each feature to be independent. The algorithm variant used here is the Bernoulli Naïve Bayes, which also assumes the features to be binary, i.e., indicating absence or presence of the word. It also has a few hyperparameters, namely:

- **Laplace smoothing parameter:** set to 1; added term that accounts for features being not present in a sample, preventing zero probabilities;
- **Binarization threshold:** set to 0; the threshold at which the feature vectors are going to be binarized. Meaning, anything equal to or below the threshold is false, and everything above is true.

The decision rule for the Bernoulli Naïve Bayes variant contains two terms which use the presence and absence of features to bump up or tone down the probability of a sample x_i towards a class y, as previously shown in Equation 2.4.

The SVM algorithm uses a linear kernel to perform classification. It is based on the LIBLINEAR[3] which optimizes the Squared-hinge loss with L_2 penalty. The original SVM algorithm optimizes the Hinge loss, but the maximum margin separation is still performed. The error penalty term C is also set to 1; a higher value enforces more strict penalties, but could lead to overfitting the data, where the model tries to adapt too tightly to the training data, instead of learning a more generalized model.

The Logistic Regression algorithm is somewhat similar to the SVM algorithm. It also depends on the LIBLINEAR solver and is set to use L_2 penalty. The LR classifier has a sigmoid function which squashes the linear transformation result on the inputs to a value between 0 and 1, which can be interpreted as the probability of belonging to a given class. Additionally, while the SVM algorithm tries to find the optimally best training samples, i.e., support vectors, to distinguish the classes, the Logistic Regression algorithm only tries to find a hyperplane separating the classes by minimizing the loss function. The penalty term C for the LR is also set to its default value of 1.0.

The Random Forest classifier is an estimator composed of multiple decision-tree classifiers, also known as ensemble. The decision tree classifiers are trained according to the CART training algorithm, which supports numerical attributes. Each subclassifier is

[1] http://scikit-learn.org
[2] https://www.python.org
[3] https://www.csie.ntu.edu.tw/ cjlin/liblinear/

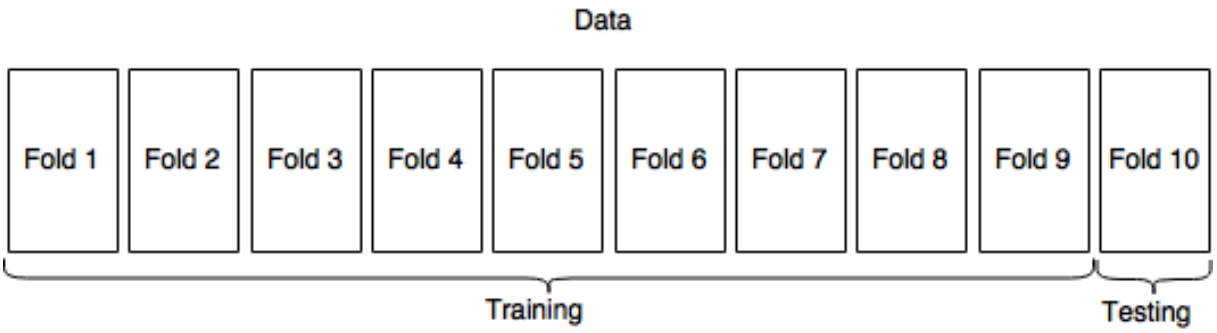

Figure 7 – Visualization of a 10-fold partitioning of the data.

Source: The author

trained using samples from the original training data, while allowing for replacement, i.e., samples can be used to train multiple classifiers. The prediction is performed by averaging the prediction of each classifier in the ensemble. The number of estimators in the ensemble is set to 200, which preliminary experiments showed to be a reasonable number.

5.4 Cross Validation

As described in Chapter 4, a few considerations regarding the split of the data are taken when dealing with the two experiment settings contemplated in this study (single-dataset and cross-dataset). The single dataset experiments are intended to evaluate features and classifiers individually on each of the datasets, while the cross-dataset experiments are intended to evaluate how much "knowledge" different learning algorithms and features can apply from one dataset to another of the same domain, or from one domain to another, e.g., movie reviews to product reviews.

For the single dataset experiments, a more concise and stable way of evaluating features and learning algorithms can be done with a technique known as cross-validation. Cross-validation allows one to have more statistic meaningful metrics by training multiple models on the same data. The common cross-validation technique known as K-fold cross-validation was used (JONES, 1987). It splits the data into K folds, and K different models are trained using one fold for testing and the remaining $K - 1$ are used for training. No samples are present in more than one fold, to avoid testing data information to leak into the training data. All the folds have equally distributed samples from each of the classes. This is also known as stratified data sample, and is done in order to ensure the testing data has a good representation of the actual training data. The Figure 7 demonstrates this visually, for better understanding.

5.5 Evaluation Metrics

In order to properly evaluate and compare different feature extraction algorithms and classifiers it is necessary, in addition to performing balanced data splits, to rely on a fair metric of comparison. This section of this chapter introduces the metrics utilized in the experiments performed in this study.

The most common evaluation metric in Machine Learning and Data Analysis is Accuracy, which is defined by the number of correct predictions over the total number of classified examples. Even though it serves as a good indicator of a model quality, it is trivial to point out a case where the accuracy could be misleading. For example, in extremely unbalanced settings, the accuracy might be high but the desired expectations might not be met. Concretely, if out of 100 samples 99 samples are from one class and only one example from the other, a model could achieve high accuracy by assigning all the examples to the majority class. A model that makes more mistakes but correctly classifies rare samples might be better than a model that makes fewer mistakes by assigning all samples to the majority class. It varies for each problem, however, how important it is to actually correctly classify the minority class, and dealing with class unbalance is a new research problem which is not in the scope of this study.

The metrics adopted in this work are Precision, Recall and F1-score (also known as F-measure or F-score). Precision and Recall are defined by Equation 5.1 and Equation 5.2 respectively.

$$Precision = \frac{tp}{tp + fp} \tag{5.1}$$

$$Recall = \frac{tp}{tp + fn} \tag{5.2}$$

Where tp, tn, fp and fn represent:

- **True positive:** samples correctly attributed to the relevant class;
- **True negative:** samples correctly attributed to the other class;
- **False positive:** samples from the other class mistakenly attributed to the relevant class;
- **False negative:** samples from the relevant class mistakenly attributed to the other class.

In the definitions above, even though it is used **positive** and **negative**, they do not strictly mean the class of a sample, i.e., in our case, they are not related to the positive

or negative label of the reviews. Instead, they depend on the perspective. More concisely, in a binary classification problem where it is intended to distinguish between cats and dogs, given some specific features about the animal, one might want to assess the model performance for both cats and dogs individually. From the perspective of evaluation of the class "dog", "dog" is the relevant (positive) class and "cat" is the other class, and vice-versa.

The F1-score is a metric of test accuracy for binary classification problems given by the harmonic-mean of the Precision and Recall as illustrated by Equation 5.3.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.3}$$

Another way to describe the above metrics using an intuitive understanding of what these formulas actually mean is as follows:

- **Precision:** of all the instances attributed to the relevant class, how many were actually correct? This metric depicts the capability of a model correctly classifying instances to the relevant class.

- **Recall:** of all the relevant instances, how many were correctly identified, i.e., attributed to the positive class? This metric depicts the model's capability of retrieving all the relevant instances.

The F1-score can be understood as a weighted average of these two metrics. The higher the metric is, the better the model, being a score of 1 a perfect model, and a score of 0 the worst model.

Since the problem in this study involves two classes that are equally important to look at, the F1-score is computed individually for each of them. In order to summarize both of these metrics into an overall metric for the model, taking the F1-scores for both classes into account, there are two commonly used alternatives: micro-average and macro-average.

The micro-average F1-score is determined by jointly calculating the precision and recall scores by adding the true/false positives and negatives for each class before performing the harmonic mean. The macro-average is simply the mean of the F1-score for both classes. The micro-average presents an average over instances, while the macro-average presents an average over classes. Since a good overall performance for both classes is the most important factor in consideration, the macro-average is more appropriate and is the main metric of choice for the comparisons and analyses in this work.

For the experiment involving cross-validation, mean and standard deviation of the macro-averaged F1-score across all folds is taken into account for evaluation.

5.6 Experimental Results

Knowing the experimental framework and its details, and the setup of the experiments contemplated in this study, this section aims at answering the proposed research questions by performing analyses from the observed results.

5.6.1 Evaluating Feature Extraction methods

This analysis takes into account the evaluation results from the experiments covered by the cross-validation evaluation of different features across the 4 classifiers and 5 datasets (individually), in order to better isolate the impact of using different features. The Doc2Vec is not included in this analysis because gensim's[4] API, which was used to train the Doc2Vec models, is not compatible with the scikit-learn cross-validation API.

From Table 3, it is possible to notice that count-based features generally outperform the word-embeddings; notably, the TF-IDF-weighted variant of the N-gram showing some improvements over the pure N-gram. The exceptions being the Word2Vec feature, however, stands out between the word-embeddings, and the Bigram feature which yields poorer results compared to the other count-based features. Word-embeddings are commonly used in many other NLP-related tasks and other text classification problems other than sentiment classification. It is possible that the embedding generation algorithm, where word's neighborhoods are used to predict other words, could lead to sentiment-expressing words clustered together. More concretely, words such as "good" and "bad" often appear in similarly structured sentences (e.g. "this movie is [good/bad]", meaning they are usually accompanied by the same words. Therefore, they have similar dense representations in the word embedding. Another possibility is that the mean vector of all the words in a document misses out on some information which other alternatives, such as different document vector generation schemes or other algorithms, might be able to catch.

There's also difference in performance between word-embedding-based features, meaning some embedding generation schemes might be better suited for different tasks, since, for example, the Numberbatch embedding - engineered from multiple embeddings to achieve better results in other NLP tasks - actually still does not beat Word2Vec.

It is important to notice, however, that the AffectiveSpace depends on a "concept parser", which parses sentences and extracts lemmatized (shorter base form of a word) n-grams of words that are not necessarily in a neighborhood of each other, but might have a syntactic relation, even though they are appear distant in the sentence. However, the publicly available version of the concept parser[5] has some missing files, dependencies, and has inconsistencies with the online API[6] and the word entries on the embedding space.

[4] https://radimrehurek.com/gensim/
[5] https://sentic.net/parser.zip
[6] http://sentic.net/demos/#concept

It is very likely that most of the observed under-performance is due to the attempted reproduction of the concept parser missing out on various details.

Feature	F1-Score	std
tfidf	0.7855	0.0205
ngram	0.7798	0.0215
unigram	0.7763	0.0215
w2v	0.7637	0.0214
glove	0.7406	0.0199
numberbatch	0.7339	0.0216
bigram	0.7280	0.0189
dependency	0.6767	0.0164
affectivespace	0.5854	0.0160

Table 3 – Mean Cross-validation F1-Score and standard deviation across all features and datasets.

5.6.2 Evaluating machine learning algorithms

From Figure 8 it's clear that the SVM performs slightly better when compared to the other algorithms. Nevertheless, Logistic Regression also performs very well, sometimes even achieving better results. This is to be expected, as both algorithms are closely linked (SWERSKY, 2014).

The Random Forest and Naïve Bayes classifiers are outperformed by the other two classifiers. Since the RF is an ensemble of classifiers, it might be surprising that it actually does not perform better than the other alternatives. One possible explanation is that its higher learning capability might have led to overfitting, and some further regularization (e.g. pruning) must be used to alleviate this. It might be possible to achieve better results by performing a search through the various hyperparameters of the algorithm, such as the number of estimators, i.e., decision trees in the ensemble. As for the NB algorithm, it is possible the model's assumption about the problem, i.e., no dependence between features, might negatively affect its performance too much. Another detail about the Bernoulli variant of the NB is that it binarizes input vectors. This causes the TF-IDF extracted features, which is a weighted N-gram, to be exactly the same as the N-gram features.

From Figure 9 it can be noticed that a common trend between all classifiers is that they usually perform better on the larger datasets (the ones containing more documents), the notable exception being SST, which despite being based on RT and being roughly a thousand documents smaller, still shows better results on all algorithms. It is possible that the additional preprocessing and labeling done by (SOCHER; PERELYGIN; WU, 2013) and removal of some samples has contributed for making it a cleaner and less noisy dataset.

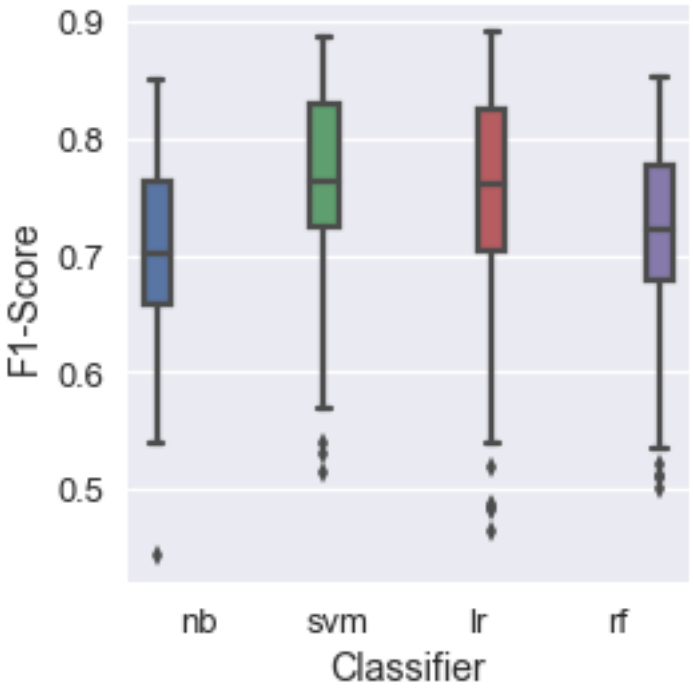

Figure 8 – Visualization of cross-validation F1-Scores across all features and datasets.

Figure 9 – Median cross-validation F1-Scores across all features, shown for datasets of increasing total number of documents.

5.6.3 Effect of preprocessing on count-based features

The count-based features were evaluated both with and without additional preprocessing. The objective here is to investigate on whether ignoring stopwords and not taking capitalization into account, by lowercasing all the words, affects the results.

Even though it is common practice for other NLP-related and text classification tasks, this kind of pre-processing has a negative impact on all features, as shown in Table 4. A very likely hypothesis for that phenomenon is due to the removal of negating words such as "not", or adverbs such as "very" and verbs "do". The Bigram feature is the most affected of all features, possibly because common bigrams such as "not good", "not bad", "do not" aren't possible when these stop words are removed, and those could be - as it is for humans - a source of significant information regarding the sentiment of a sentence.

Feature	F1-Score	F1-Score (pre)	Change
bigram	0.7280	0.6202	-0.1078
tfidf	0.7855	0.7751	-0.0104
ngram	0.7798	0.7698	-0.0100
unigram	0.7763	0.7680	-0.0083

Table 4 – Effect of preprocessing on count-based features when compared across all datasets and classifiers.

5.6.4 Cross-Corpus Evaluation

Another important property of SA systems - and ML systems in general - is how well they perform on completely new data. Not only on held out data from the same corpus, but also when entirely new data with different document lengths, vocabulary, etc. This effect is evaluated in two different settings: new corpus with same domain, and new corpus of a different domain. Despite being of different domains, the problem under analysis is still the same (sentiment analysis), and there's valuable knowledge to be obtained.

5.6.4.1 Same domain

This section discusses the portion of this evaluation related to the experiments on learning from a new corpus of the same domain.

For this evaluation, two datasets of one domain were used at a time, one as training data and the other as testing data, for all dataset pairs with the exception of the SST and RT datasets, since they are based on the same corpus and it would not make sense to evaluate them together.

From Table 5, it is clear to see that, in general, sparse features still slightly outperform dense features. Although sometimes the embedding features produce better

feature	accuracy		precision		recall		fmeasure	
	mean	std	mean	std	mean	std	mean	std
tfidf	0.6887	0.1019	0.7648	0.1444	0.7420	0.2163	0.7124	0.1471
ngram	0.6840	0.0988	0.7711	0.1429	0.7171	0.2165	0.7024	0.1448
unigram	0.6781	0.0999	0.7716	0.1413	0.7065	0.2245	0.6949	0.1435
numberbatch	0.6911	0.1108	0.7779	0.1283	0.6933	0.2444	0.6919	0.1547
glove	0.6927	0.0867	0.7810	0.1313	0.6839	0.2301	0.6907	0.1384
dependency	0.6646	0.0926	0.7292	0.1280	0.7134	0.2110	0.6885	0.1166
d2v	0.6691	0.1108	0.7448	0.1198	0.6920	0.2135	0.6883	0.1371
w2v	0.6936	0.1136	0.7981	0.1380	0.6882	0.2776	0.6800	0.1974
bigram	0.6533	0.0802	0.7358	0.1230	0.6611	0.2174	0.6628	0.1382
affectivespace	0.6169	0.0783	0.6374	0.0891	0.6977	0.1756	0.6575	0.1137

Table 5 – Feature Extractors' metrics when evaluating on a new corpus of the **same domain.**

results, the count-based features seem to be more stable. TF-IDF and Word2Vec come out on top between their group of features.

5.6.4.2 Different domain

This section discusses the results of the experiments on learning from a new corpus from a different domain (cross-domain).

For this experiment, two datasets from one domain were used as training data, while other two datasets from different domains, regarding the training ones, were used for assessing the results. The RT dataset was not considered in this setting, and SST - with its improved reannotation - was used instead. It is important to notice that both datasets from the product reviews domain are from the same source (Amazon), but they were collected at very distinct moments and have very different class balance and document lengths, so that this evaluation is still relevant.

According to Table 6, it is clear that, in general, for this setting the word-embedding-based features perform more consistently than the count-based features. Word2Vec and TF-IDF still yielded better results; Furthermore, Word2Vec yields, in general, better results than all of the count-based features. A positive aspect of the WE features is that they don't rely on vocabulary as much as count-based. Count-based features are entirely dependant on the words found in the training samples, not being able to make predictions based on unseen words. Whereas, for example, an unseen word during training can be present in the WE vocabulary and have a close representation to another word that was present during the learning phase.

Overall performances are higher in the cross-domain setting when compared to the same-domain, possibly due to the increased amount of training data, since both datasets

feature	accuracy		precision		recall		fmeasure	
	mean	std	mean	std	mean	std	mean	std
d2v	0.6578	0.0968	0.6996	0.1361	0.8232	0.1083	0.7412	0.0684
w2v	0.6801	0.0764	0.7627	0.1228	0.7245	0.1570	0.7232	0.0881
tfidf	0.6603	0.0880	0.6701	0.2242	0.7601	0.2344	0.6984	0.1998
glove	0.6409	0.0958	0.7407	0.1410	0.7131	0.1895	0.6943	0.1059
unigram	0.6523	0.0846	0.6704	0.2229	0.7416	0.2245	0.6908	0.1950
ngram	0.6573	0.0878	0.6742	0.2217	0.7259	0.2268	0.6878	0.1993
numberbatch	0.6304	0.0953	0.7271	0.1516	0.7225	0.2197	0.6861	0.1227
bigram	0.6298	0.0720	0.6518	0.2157	0.7148	0.2239	0.6690	0.1923
affectivespace	0.5499	0.0411	0.6392	0.1350	0.7204	0.1960	0.6473	0.0695
dependency	0.5278	0.0873	0.6936	0.1740	0.6432	0.3114	0.5861	0.1422

Table 6 – Feature Extractors' metrics when evaluating on a new corpus of a **different domain.**

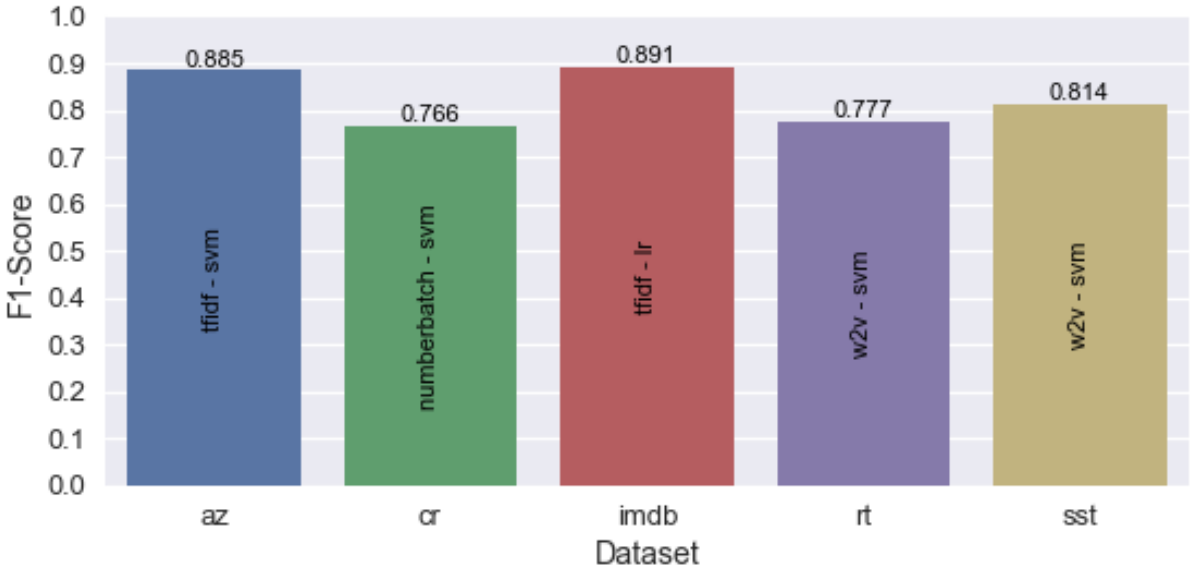

Figure 10 – Cross-validation F1-Score for the best combination of feature and classifier on each dataset.

Source: The author

from one domain are used in conjunction.

5.6.5 Best setting per dataset

Finally, the Figure 10 shows the setting (combination of feature and classifier) with the best result from the single-dataset cross-validation experiment. Both the TFIDF and Word2Vec features, and the SVM and LR classifiers are the most common among the best results, which showed to be a recurring trend observed in our previous analyses.

6 Conclusion and Future Works

This chapter summarizes and discusses the contributions, shortcomings and open paths of exploration for further improvement of the present research work.

This work has presented an assessment study of the polarity classification, a subtask in Sentiment Analysis.

Taking the polarity classification problem in SA as a study case, this study was intended for evaluating the impact of the major components responsible for feature extraction and training/testing the selected supervised machine learning algorithms. A total of five well-known benchmark polarity classification datasets from two domains were employed.

For facilitating the evaluation of different learning scenarios and experimental settings, this work provided an implementation of the adopted assessment methodology. This implementation is available as several Python scripts which provide, to the user, several services concerning feature extraction, machine learning techniques, and two evaluation strategies (cross-corpus and cross-domain) using several datasets.

Finally, the comparative analysis of both sparse and dense document representations, learning algorithms, preprocessing, and learning from different datasets were also provided. The analyses were followed by discussions of the obtained results. Furthermore, this study highlighted important discussions to the field regarding the choice of feature for cross-domain learning in sentiment polarity classification, which is still - to the best of our knowledge - largely unexplored.

Future Works. Despite the experimental contributions to the task of polarity classification achieved by this work, several questions remain unanswered.

With the recent trend of Deep Learning (DL) - a broader family of machine learning methods based on deep structured neural networks (SCHMIDHUBER, 2015) - many areas of study including NLP, Computer Vision, and Voice Recognition are now shifting towards such deep neural networks (DNN). Particularly for the NLP field, DNN models have been proposed and are commonly used in conjunction with word embeddings. Following the above trend, another line of investigation will extend this work to include such recent DNN advancements.

For evaluating word embeddings, a single document vector generation scheme was employed: the mean vector of all of the words in the document. Although demonstrating competitive results to the count-based features, this is not the only possible way of representing a document vector. Other possible techniques can use weighted means, or the

maximum value, for instance. In addition, while Doc2Vec can be considered one of them, there are more sophisticated WE learned using larger chunks of texts, such as "attention mechanisms" (LUONG; PHAM; MANNING, 2015) which allow a model to learn which parts of a document are more important to focus on.

Feature selection and combination, and classifier optimation by tunning their hypermeters (e.g. grid search or random search (BERGSTRA; BENGIO, 2012)) are also techniques which might lead to improved overall performance, but were not covered by this study. An analysis of how they affect the proposed evaluation scenarios would be a promising path towards improvement of the results reported here.

Additionally, for the preprocessing impact analysis, further improvements can be made on the stopword removal scheme, allowing a more flexible selection of words to be removed, as well as further inspection of the words the cause the most impact on classification when removed. Other preprocessing steps could also be analyzed, such as stemming or lemmatization, which are techniques to normalize words to a common base form. For that, the application of an optimization approach, such as genetic algorithms, can be exploited in this case.

Different languages have different syntactical structures. Extending our evaluations to multilingual setups in conjunction with the other aspects covered in this study would also yield valuable insights into knowledge.

The ever-increasing shift towards DNN in many areas might be a clear indicative of the future path for the study here established. It can be used as a stable ground for evaluation of newer features, learning algorithms and other related techniques, not only for the problem of Sentiment Analysis, but for text classification in general.

Bibliography

BAKER, L. D.; MCCALLUM, A. K. Distributional clustering of words for text classification. In: ACM. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. [S.l.], 1998. p. 96–103. 22

BENGIO, Y. et al. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, v. 3, p. 1137–1155, 2003. ISSN 15324435. 19, 22

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, JMLR.org, v. 13, p. 281–305, fev. 2012. 52

BESPALOV, D. et al. Sentiment classification with supervised sequence embedding. In: *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*. Berlin, Heidelberg: Springer-Verlag, 2012. (ECML PKDD'12), p. 159–174. 18

BOLLEN, J.; MAO, H.; ZENG, X. Twitter mood predicts the stock market. *Journal of Computational Science*, Elsevier, v. 2, n. 1, p. 1–8, 2011. 18

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: ACM. *Proceedings of the fifth annual workshop on Computational learning theory*. [S.l.], 1992. p. 144–152. 25

BREIMAN, L. et al. *Classification and regression trees*. [S.l.]: CRC press, 1984. 26

BURNS, N. et al. Sentiment analysis of customer reviews: Balanced versus unbalanced datasets. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 6881 LNAI, n. PART 1, p. 161–170, 2011. ISSN 03029743. 30

CAMACHO-COLLADOS, J.; PILEHVAR, M. T. On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis. n. 2015, 2017. Disponível em: <http://arxiv.org/abs/1707.01780>. 30

CISCO. *Forecast and Methodology, 2014-2019 White Paper*. [S.l.], 2015. 17

CLASTER, A. *Inside the Obama Analytics Cave*. 2016. Disponível em: <http://www.sayreforchair.com/wp-content/uploads/2016/03/Inside-the-Cave-Obama-Analytics1.pdf>. 17

COX, D. R. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, JSTOR, p. 215–242, 1958. 25

HU, M.; LIU, B. Mining and summarizing customer reviews. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining KDD 04*, v. 04, p. 168, 2004. ISSN 1581138889. Disponível em: <http://portal.acm.org/citation.cfm?doid=1014052.1014073>. 40

JACKSON, P. Introduction to expert systems. Addison-Wesley Pub. Co., Reading, MA, 1986. 28

JO, Y. Aspect and Sentiment Unification Model for Online Review Analysis. *Proceedings of the fourth ACM international conference on Web search and data mining*, p. 815–824, 2011. Disponível em: <http://uilab.kaist.ac.kr/research/WSDM11/wsdm400-jo.pdf>. 40

JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, MCB UP Ltd, v. 28, n. 1, p. 11–21, 1972. 21

JONES, L. *The Collected Works of John W. Tukey: Philosophy and Principles of Data Analysis 1965-1986*. [S.l.]: CRC Press, 1987. v. 4. 639 p. 35, 42

KHARDE, V.; SONAWANE, S. et al. Sentiment analysis of twitter data: A survey of techniques. *arXiv preprint arXiv:1601.06971*, 2016. 18, 19

KURIAN, N. Cross Domain Sentiment Classification : Current Solutions. v. 3, n. 5, p. 3–6, 2014. 30

LE, Q. V.; MIKOLOV, T. Distributed Representations of Sentences and Documents. v. 32, 2014. ISSN 10495258. Disponível em: <http://arxiv.org/abs/1405.4053>. 24

LEOPOLD, E.; KINDERMANN, J. Text Categorization with Support Vector Machines: How to Represent Text in Input Space? *Machine Learning*, v. 46, n. 3, p. 423–444, 2002. ISSN 0885-6125. 29

LEVY, O.; GOLDBERG, Y. Dependency based word embeddings. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, v. 2, p. 302–308, 2014. 23

LIU, B. *Sentiment analysis and opinion mining*. [s.n.], 2012. ISBN 9781608458844. Disponível em: <https://scholar.google.co.in/scholar?q=related:McjgThBIKwgJ:scholar.google.com/{&}hl=en{&}as{_}>. 17, 18

LUONG, M.-T.; PHAM, H.; MANNING, C. D. Effective Approaches to Attention-based Neural Machine Translation. *Emnlp*, n. September, p. 11, 2015. ISSN 10495258. Disponível em: <http://arxiv.org/abs/1508.04025>. 52

MAAS, A. L. et al. Learning Word Vectors for Sentiment Analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, p. 142–150, 2011. 40

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. Text classification and Naive Bayes. *Introduction to Information Retrieval*, n. c, p. 260, 2008. Disponível em: <https://books.google.co.th/books/about/Introduction{_}to{_}Information{_}Retrieval.html?id=GNvtngEACA>. 25, 30

MATSUOKA, J.; LEPAGE, Y. Measuring Similarity from Word Pair Matrices with Syntagmatic and Paradigmatic Associations. *Proceedings of the 4th Workshop on Cognitive Aspects of the Lexicon*, p. 77–86, 2014. 27

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, Faculty of Engineering, Ain Shams University, v. 5, n. 4, p. 1093–1113, 2014. ISSN 20904479. Disponível em: <http://dx.doi.org/10.1016/j.asej.2014.04.011>. 18, 28, 29, 30

MIKOLOV, T. Efficient Estimation of Word Representations in Vector Space. *IJCAI International Joint Conference on Artificial Intelligence*, v. 2015-Janua, p. 4069–4076, 2013. ISSN 10450823. Disponível em: <http://arxiv.org/pdf/1301.3781v3.pdf>. 22

MILLER, G. Wordnet. v. 38, n. 11, p. 39–41, 1995. ISSN 00010782. 28

PANG, B.; LEE, L. A Sentimental Education: Sentiment Analysis using Subjectivity Summation based on Minimum Cuts. *ACL '04 Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, p. 271, 2004. ISSN 1554-0669. 29

PANG, B.; LEE, L. Seeing stars. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*, n. 1, p. 115–124, 2005. 39

PANG, B.; LEE, L.; VAITHYANATHAN, S. Thumbs up?: sentiment classification using machine learning techniques. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, p. 79–86, 2002. ISSN 1554-0669. Disponível em: <http://portal.acm.org/citation.cfm?id=1118693.1118704>. 17

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. 41

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, p. 1532–1543, 2014. ISSN 10495258. 23

PORIA, S. et al. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. *Knowledge-Based Systems*, Elsevier B.V., v. 69, n. 1, p. 45–63, 2014. ISSN 09507051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2014.05.005>. 23, 30

QIU, G. et al. DASA: Dissatisfaction-oriented Advertising based on Sentiment Analysis. *Expert Systems with Applications*, Elsevier Ltd, v. 37, n. 9, p. 6182–6191, 2010. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2010.02.109>. 29

RAMBOCAS, M.; GAMA, J. et al. *Marketing research: The role of sentiment analysis*. 2013. 18

RAVI, K.; RAVI, V. A survey on opinion mining and sentiment analysis: Tasks, approaches and applications. *Knowledge-Based Systems*, Elsevier B.V., v. 89, p. 14–46, 2015. ISSN 09507051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2015.06.015>. 17, 18, 30

ROJAS-BARAHONA, L. M. Deep learning for sentiment analysis. *Language and Linguistics Compass*, v. 10, n. 12, p. 701–719, 2016. ISSN 1749818X. Disponível em: <http://doi.wiley.com/10.1111/lnc3.12228>. 30

ROTHFELS, J. Unsupervised sentiment classification of English movie reviews using automatic selection of positive and negative sentiment items. *Strategy*, 2010. 28

Rushdi Saleh, M. et al. Experiments with SVM to classify opinions in different domains. *Expert Systems with Applications*, Elsevier Ltd, v. 38, n. 12, p. 14799–14804, 2011. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2011.05.070>. 30

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, v. 61, p. 85 – 117, 2015. 51

SCHNABEL, T. et al. Evaluation methods for unsupervised word embeddings. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, n. September, p. 298–307, 2015. 31

SERRANO-GUERRERO, J. et al. Sentiment analysis: A review and comparative analysis of web services. *Information Sciences*, Elsevier, v. 311, p. 18–38, 2015. 18

SOCHER, R. et al. Parsing natural scenes and natural language with recursive neural networks. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. USA: Omnipress, 2011. (ICML'11), p. 129–136. 19

SOCHER, R.; PERELYGIN, A.; WU, J. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the ...*, p. 1631–1642, 2013. ISSN 1932-6203. 39, 46

SPEER, R.; CHIN, J.; HAVASI, C. {ConceptNet 5.5: A}n Open Multilingual Graph of General Knowledge. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, n. Singh 2002, p. 4444–4451, 2017. Disponível em: <http://arxiv.org/abs/1612.03975>. 23

SWERSKY, K. *Support Vector Machines vs Logistic Regression*. 2014. Disponível em: <http://www.cs.toronto.edu/~kswersky/wp-content/uploads/svm_vs_lr.pdf>. 46

TANG, D.; QIN, B.; LIU, T. Deep learning for sentiment analysis: Successful approaches and future challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, v. 5, n. 6, p. 292–303, 2015. ISSN 19424795. 30

TENSORFLOW Word2Vec tutorial. 2016. Disponível em: <https://www.tensorflow.org/tutorials/word2vec>. 23

TURNEY, P. D. Thumbs up or thumbs down? Semantic Orientation applied to Unsupervised Classification of Reviews. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, n. July, p. 417–424, 2002. ISSN 0738467X. 17, 29

TURNEY, P. D.; PANTEL, P. From frequency to meaning: Vector space models of semantics. *CoRR*, abs/1003.1141, 2010. 19

TWITTER. *Twitter, Inc. Common Stock*. 2013. Disponível em: <http://www.sec.gov/Archives/edgar/data/1418091/000119312513390321/d564001ds1.htm#toc564001_1>. 17

WEI, W.; GULLA, J. A.; FU, Z. Enhancing negation-aware sentiment classification on product reviews via multi-unigram feature generation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 6215 LNCS, p. 380–391, 2010. ISSN 03029743. 30

WIEBE, J. Learning subjective adjectives from corpora. *Proceedings of the National Conference on Artificial Intelligence*, n. 1, p. 735–741, 2000. Disponível em: <http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Learning+subjective+adjectives+from+cor>. 17

WIEBE, J. M.; BRUCE, R. F.; O'HARA, T. P. Development and use of a gold standard data set for subjectivity classifications. *Proceedings of the Association for Computational Linguistics*, p. 246–253, 1999. 27

WU, Q.; TAN, S. A two-stage framework for cross-domain sentiment classification. *Expert Systems with Applications*, Elsevier Ltd, v. 38, n. 11, p. 14269–14275, 2011. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2011.04.240>. 30