PROYECTO FIN DE CARRERA

Título:	Prototipo de un Sistema de Análisis de Sentimientos Basado en Algoritmos de Ensemble para la Combinación de Técnicas de Aprendizaje Automático Profundas y Superficiales
Título (inglés):	Prototype of a Sentiment Analysis System Based on Ensemble Algorithms for Combining Deep and Surface Machine Learning Techniques
Autor:	Óscar Araque Iborra
Tutor:	J. Fernando Sánchez Rada
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	Tomás Robles
Vocal:	Mercedes Garijo
Secretario:	Joaquín Salvachúa
Suplente:	Francisco González

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE MÁSTER

Prototype of a Sentiment Analysis System Based on Ensemble Algorithms for Combining Deep and Surface Machine Learning Techniques

Óscar Araque Iborra

Junio de 2016

Resumen

Las técnicas de aprendizaje profundo para el Análisis de Sentimientos se han vuelto muy populares. Estas técnicas proveen de capacidades de extracción de características automáticas, así como representaciones más ricas y de mejores prestaciones que las técnicas basadas en características tradicionales. Estos enfoques superficiales y tradicionales están basados en características complejas extraídas manualmente, siendo este proceso de extracción una pregunta fundamental en los métodos basados en aprendizaje a partir de características. Dichos enfoques largamente establecidos pueden producir sólidos sistemas de referencia, al igual que sus capacidades de predicción pueden ser usadas en conjunto con los métodos incipientes basados en aprendizaje profundo.

Este trabajo de fin de máster busca mejorar el rendimiento de las nuevas técnicas de aprendizaje profundo, integrándolas con enfoques superficiales basados en características extraídas manualmente. Las contribuciones de este trabajo de fin de máster son siete. Primera, hemos desarrollado un clasificador de sentimientos basado en aprendizaje automático usando un modelo de *word embeddings* y un algoritmo de aprendizaje automático lineal. Este clasificador nos sirve como referencia con la que comparar resultados posteriores. Segunda, proponemos dos técnicas de *ensemble* que agregan nuestro clasificador de referencia con otros clasificadores ampliamente usados en el campo del Análisis de Sentimientos. Tercera, también proponemos dos modelos que combinan tanto características superficiales como profundas con el objetivo de unir información proveniente de distintas fuentes. Como cuarta contribución, introducimos una taxonomía que clasifica los distintos modelos encontrados en la literatura, al igual que los modelos que proponemos. Quinta, llevamos a cabo varios experimentos que comparan el rendimiento de estos modelos en comparación con la referencia de aprendizaje profundo. Para llevar esto a cabo, empleamos cinco conjuntos de datos públicos que fueron extraídos del dominio del *microblogging*. Sexta, como resultado, un estudio estadístico confirma que el rendimiento de los modelos propuestos supera el de la referencia original en *F1-Score*. Finalmente, un caso de estudio adicional se ha llevado a cabo con la finalidad de aumentar el alcance de la evaluación de los modelos propuestos. Este caso de estudio introduce dos grandes cambios en la evaluación. Por un lado, el dominio es distinto, ya que el análisis se realiza en el dominio de las críticas de usuarios. Por

otro lado, la granularidad con la que el análisis se realiza es mayor, siendo éste el nivel de análisis de sentimientos basado en aspectos.

Palabras clave: Combinación de clasificadores, Aprendizaje Profundo, Análisis de Sentimientos, Aprendizaje Automático, Procesado de Lenguaje Natural

Abstract

Deep learning techniques for Sentiment Analysis have become very popular. They provide automatic feature extraction and both richer representation capabilities and better performance than traditional feature based techniques. Traditional surface approaches are based on complex manually extracted features, and this extraction process is a fundamental question in feature driven methods. These long-established approaches can yield strong baselines on their own, and their predictive capabilities can be used in conjunction with the arising deep learning methods.

This master thesis seeks to improve the performance of new deep learning techniques integrating them with traditional surface approaches based on manually extracted features. The contributions of this master thesis are seven-fold. First, we develop a deep learning based sentiment classifier using a word embeddings model and a linear machine learning algorithm. This classifier serves us as a baseline with which we can compare subsequent results. Second, we propose two ensemble techniques which aggregate our baseline classifier with other surface classifiers widely used in the field of Sentiment Analysis. Third, we also propose two models for combining both surface and deep features to merge information from several sources. As fourth contribution, we introduce a taxonomy for classifying the different models found in the literature, as well as the ones we propose. Fifth, we conduct several experiments to compare the performance of these models with the deep learning baseline. For this, we employ five public datasets that were extracted from the microblogging domain. Sixth, as a result, a statistical study confirms that the performance of these proposed models surpasses that of our original baseline on F1-Score. Finally, an additional case study is developed in order to broaden the scope of the proposed models evaluation. This case study introduces two major changes in the evaluation. On the one hand, the domain is different, making the analysis on the review domain. On the other hand, the granularity of the analysis is increased, as it is performed at the aspect based sentiment analysis level.

Keywords: Ensemble, Deep Learning, Sentiment Analysis, Machine Learning, Natural Language Processing

Agradecimientos

El desarrollo de este trabajo de fin de máster ha sido un camino largo, que probablemente no hubiera sido posible sin mucha gente que me ha apoyado, a veces de manera sorprendentemente incansable.

Quiero dar gracias a mis padres, que desde que comencé la carrera (hace ya seis largos años) siempre me han apoyado y me han animado a seguir. Nunca han dejado de creer en mí, y siempre han sido una fuente de confianza a la hora de enfrentarme a los distintos retos que se me han presentado a lo largo de este satisfactorio periodo. En conjunto a mis padres se encuentran muchos miembros de mi familia, que han estado ahí en caso de necesidad. Mis abuelas, tíos y tías, etcétera.

Sin duda Cristina se merece una mención especial en estas páginas, ya que ha tenido que escuchar muchas horas de eternas dudas, disyuntivas acerca del futuro y devaríos varios. Gracias por estar siempre ahí, y darme ese empujoncito que a veces necesitaba.

Por supuesto, doy las gracias a Carlos Ángel, cuya perseverancia ha ayudado en gran parte al desarrollo del trabajo, y finalmente a llevarlo a buen puerto. Sin su guía no habría llegado donde he llegado.

También quiero dar gracias a mis compañeros de laboratorio. A J, que me ha ayudado a lo largo de todo este año, aconsejándome y guiándome donde yo poco o nada sabía: gracias por dedicarme ese tiempo. A Álvaro, que ha sido un tercer tutor. Y a todos los compañeros (tanto del GSI como de clase) que, de alguna manera u otra, han compartido este viaje conmigo: Silvia, Adrián, Iván, Verónica, Jose, Pablo, Carlos, Nacho, Constan, Enrique, Ganggao, Shengjing, y muchos más.

A todos vosotros, ¡gracias!

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This Chapter introduces the concept of sentiment analysis, one of the most important tasks in Natural Language Processing, providing a brief introduction of its context in both the academic and business environments. Given the context of the sentiment analysis problem and its main approaches, the goal of this master thesis is drafted. As part of this goal, we propose three questions that will be answered during the development of this work.

Sentiment analysis deals with the computational analysis of human opinions, sentiments, attitudes and emotions towards entities and their attributed expressed in text. The entities are very varied, and can be products, services, organizations, topics, events and their attributes.

This field have experimented a growth with those of social media on the Web -reviews, forum discussions, blog, microblogs, social networks-, because for the first time in human history, we have a huge volume of data with opinion content in digital form, making possible the advance of the field.

In fact, there was almost no research on sentiment analysis from either the linguistics or the Natural Language Processing (NLP) community before the year 2000 (1). This is partly because of the lack of opinionated text data in digital form. With the growth of the web, since early 2000, sentiment analysis has grown to be one of the most important fields of Natural Language Processing (NLP), and it is also widely studied in data mining, text mining and web mining. Besides, it has spread from computer science to social sciences and management sciences due to its importance to business and society as a whole. In the industrial environment, sentiment analysis has also thrived, with many startups emerging. To sum up, sentiment analysis has found its applications in almost every social domain and business (2).

The growth of user-generated content in web sites and social networks, such as Twitter, Amazon, and Trip Advisor, has lead to an increasing power of social networks for expressing opinions about services, products or events, among others. In this context, many Natural Language Processing (NLP) tasks are being used in order to analyze this massive information.

Sentiment analysis is very interesting to businesses and organizations because they always want to find about consumer of public opinions about their products and services. It is also relevant to government, as for example, when they have the necessity of finding public opinions about policies. Besides businesses, governments and organizations, individual consumers also want to know the opinions of other about products, services, and event political candidates before purchasing the products, using the services and making a political decision.

Opinionated data not only exists on the web, but many organizations have internal data, such as customer feedback collected from e-mails, internals polls or surveys conducted by the organizations. It is usually very important to analyze both kinds of data in order to correctly summarize customer opinions.

In recent years, hundreds of companies have appeared that work in sentiment analysis, both start-ups and established corporations, that have build or are in the process of building their own solutions (1). Added to these companies, applications are also widespread in government agencies. These agencies use social media to discover public opinions and citizen concerns. Externally, intelligent services find about issues being discussed in the social debate of other countries by monitoring the social media of these countries.

Apart from this applications, sentiment analysis is also very present in the research community, with many papers published regarding this field. Sentiment analysis techniques have been used successfully in many contexts (1). For example, using sentiment information to predict success of films and box-office revenue. Also, sentiment analysis was proved to correlate with presidential approval, in the context of political elections. All this analysis were made using a great variety of data sources, such as Twitter, blogs and news articles. Another popular area in sentiment analysis is market prediction. Sentiment analysis has been used to predict the movement of stock market indices, finding interesting correlations on some of them.

These classical applications aside, sentiment analysis has been also used to rank products and merchants through product reviews. Also, certain relationships between the National Football League betting data and public opinions in blogs and on Twitter were studied, as well sentiment flow patterns. Besides, sentiment analysis has been used to characterize social interactions.

The dominant approaches in sentiment analysis are based on *machine learning* techniques (3), (4), (5). The more traditional approach consist on combining the Bag Of Words (BOW) model with some other features, such as the syntactical analysis of the text (Sec. 2.1). Besides, normally this traditional approaches of sentiment analysis involve the use of a *sentiment lexicon* as source of sentiment knowledge (6), and the addition of this data to the previously mentioned features.

Generally, these techniques require complex hand-crafted features, being the process of obtaining the features a work intensive one. Besides, these techniques also need for very domain dependent datasets as lexicons, as many words can be either positive or negative, depending on the context they are used. As it is true that these characteristics are usually difficult to surpass, they also provide the resulting models with an interesting high specialization capacity, as well as the need for relatively low amounts of data.

As an alternative of the traditional approaches, some deep learning algorithms have shown excellent performance results in NLP tasks including Sentiment Analysis (7). In these deep learning techniques, the main idea is to learn complex features extracted from data

with minimum external contribution (8) using deep neural networks (9). These algorithms do not need to be passed manually crafted features from the data: they learn automatically complex features by themselves. Nevertheless, a characteristic feature of deep learning approaches is that they need large amounts of data to perform well (10). Both automatic feature extraction and availability of resources are very important when comparing the traditional machine learning approach and these deep learning techniques.

However, it is not clear that the domain specialization capacity of traditional approaches can be surpassed with the generalization capacity of deep learning based models, or if it is possible to successfully combine these two techniques in a wide range of applications.

Goal

In this master thesis, we seek to improve the existing deep learning methods by combining some of this deep techniques and traditional approaches. In this way, our main objective is to augment the available information that the sentiment analysis models have, and verify that this data augmentation effectively improves the performance of the traditional and deep techniques separately.

For achieving this objective, we propose to combine these two main sentiment analysis approaches through several ensemble models in which the information provided by many kinds of features is aggregated. In particular, this work considers an ensemble of classifiers, where several sentiment classifiers trained with different kinds of features are combined, and an ensemble of features, where the combination is made at the feature level. In order to study the complementarity of the proposed models, we use four public test datasets of the Twitter domain, and compare, through a statistical study, the results of these ensemble models in comparison to a deep learning baseline we have also developed. Besides, we present a taxonomy that classifies the models found in the literature and the ones proposed in this work.

With this work, we seek to answer the following questions, using the empirical results we have obtained as basis:

1. Which are the current ensemble approaches for deep and traditional techniques in sentiment analysis? How are they organized?

2. Can sentiment analysis performance of a deep learning based analyzer be improved when using the proposed ensemble and feature combination models?

3. Among the proposed models, which ones can be selected to improve the sentiment analysis performance of a deep learning based analyzer?

In addition to this, in this master thesis we also present a case study where the proposed sentiment analysis models are used in a different context. In this case the sentiment analysis is made in the review domain, not in the Twitter domain. Also, the sentiment analysis is targeted to aspects, and so a context detection algorithm has been included in the analysis. This subsystem is an added step in the preprocessing of the data.

These new characteristics pose some challenges that do not appear in the previous study. In order to address these new problems, some additional techniques are used.

Task description

This section describes the task of sentiment analysis. Generally, sentiment analysis research has been mainly carried out at three levels of granularity: document level, sentence level and aspect level. Here, we introduce them briefly (1).

Document level. The task at the document level consists on classifying whether a whole document expresses a positive or negative sentiment. For example, given a product review, the system must determine whether the document expresses an overall positive or negative opinion. This type of analysis assumes that each document expresses its opinion towards a single entity (e.g., a single product or service). Thus, it is not applicable to documents that compare two or more aspects or entities.

Sentence level. This next level is to determine whether each sentence in a document expresses a positive or negative (sometimes *neutral* polarity is included in the analysis) opinion. This analysis is more detailed than the previous document-level analysis, but it is also not appropriate for reviews that compare different entities in the same sentence.

Aspect level. Neither document nor sentence level analysis reveal what people like and dislike exactly. Concretely, they do not discover what each opinion is targeted to, that is, what is the target of the opinion. This level of analysis is also called *feature* level. Instead of looking at the documents or sentences, this analysis looks directly into which the opinions are, and which are their targets. Descending into the level of aspect level gives a much better understanding of the sentiment analysis problem. In this example: "*I really liked the meal, but the service was slow*', the sentence has

a positive orientation, but it is not possible to affirm that the sentence is entirely positive. In fact, we can only say that the sentence in completely positive about the *meal*, but still, it is negative regarding the *service*. In this way, the goal if this analysis level is to discover sentiments on entities or aspects. This way, a summary of opinions towards each entity can be displayed.

In this master thesis, we center our attention to sentence-level sentiment analysis from Chapters 2 to 4, and to aspect level sentiment analysis in Chapter 5.

The rest of the thesis is organized as follows. Chapter 2 shows previous work on traditional and deep learning approaches, as well as ensemble techniques. Besides, this chapter also describes the proposed taxonomy for classifying ensemble methods that merge surface and deep features. Furthermore, Chapter 3 addresses the proposed classifier and ensemble models. In Chapter 4, we describe the designed experimental setup, and also the experimental results are presented and analyzed in this Chapter. Continuing, Chapter 5 describes the motivation of the case study and its results. To finalize, Chapter 6 draws conclusions from the previous results and outlines for the future work in this regard.

CHAPTER 2

State of the Art

This Chapter offers a brief review of the main ideas and techniques that support the master thesis, as well as some of the related published works found in the literature. We first describe some traditional techniques used in sentiment analysis. After this, some insight is given on the new deep techniques, with a special emphasis on word vectors representations. Continuing, the main ensemble techniques are summarized, and some of they key characteristics are discussed. To finalize, this chapter also covers a key contribution of this master thesis: an ensemble taxonomy that classifies the models found in the literature, and also the ones proposed in this work.

The approaches tackling the sentiment analysis challenges are very diverse, and make use from a wide range of fundamental fields, such as machine learning and linguistics. Regarding this work, we can identify two main families of sentiment analysis techniques: surface and deep approaches. Surface techniques are based on superficial aspects of the natural language such as distribution of syntactical elements and appearance of certain tokens. Furthermore, deep approaches base their functioning in the semantic meaning of the text, and for that goal different text representations are used.

In this chapter, we will review the main techniques and trends that exist within sentiment analysis research. The traditional techniques are briefly reviewed, as they are relevant for this work, while the deep approaches will be more detailed, due to this work is heavily based on these. Also, other important techniques for this work are described, such as the ensemble of classifiers. Finally, it is also presented a taxonomy that classifies the different approaches found in the literature and the ones proposed in this work.

Surface approaches

Generally, sentiment analysis is considered as a text classification problem (2). Since it is a classification problem, any existing supervised learning method can be applied. In fact, the dominant approaches in sentiment analysis are based on *machine learning* techniques (3), (4), (5). Traditional approaches use a variation of the BOW model, where a document is mapped to a feature vector, and then is classified by machine learning techniques. The most basic version of BOW is the *one-hot vector*, which represents each word w as an $\mathbb{R}^{|V|\times 1}$ sparse vector with all 0s and one 1 at the index of w in the sorted vocabulary V, being $|V|$ the vocabulary size. Word vectors representing through this type of encoding would appear as follows:

$$w^{a} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ldots, v^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Although the BOW approach is simple and quite efficient, a great deal of the information from the original natural language is lost (11), word order is disrupted and syntactic structures are broken. Added to that, this representation represents each word as a completely

independent entity, not giving any notion of word similarity:

$$(w^{monkey})^T w^{wall} = (w^{monkey})^T w^{ape} = 0$$

Accordingly, features with a more extensive understanding of the natural language are used. Instead of restricting to BOW (unigrams), higher order n-grams can be used, such as bigrams or trigrams (12).

Another kind of feature that can be used is POS information, which is a basic form of syntactic analysis, as described in (13). In this last work the effectiveness of POS features is very noted, in contrast to the analysis discussed in (14) where the utility of these POS features in the microblogging domain -such as Twitter- is questioned. POS and many tasks intimately related with POS, such as dependency parsing, have been studied in depth, and continuous progress is being made. Dependency parsing represents the syntactic relationships between words in a sentence, and can be represented by a dependency tree. Figure 2.1 shows an example of a result of dependency parsing and POS tagging [1]. In (15), a POS tagger and dependency parser is described that surpasses the previous state-of-the-art is presented, and is based on a feed-forward neural network.

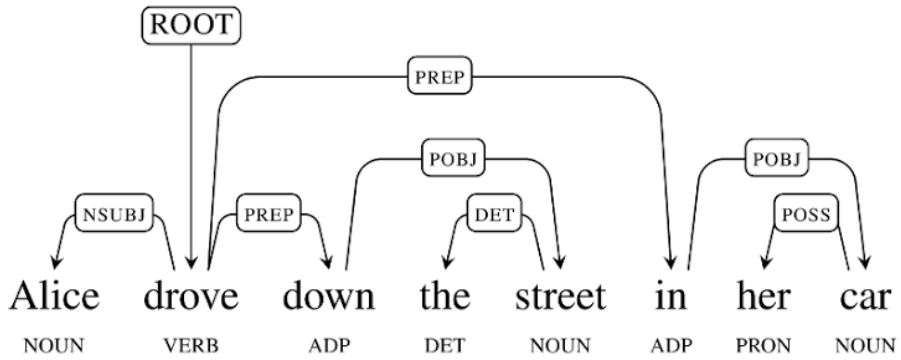


Figure 2.1: Example of dependency parsing and POS tagging.

Added to this, many machine learning techniques involve gathering a *sentiment lexicon* as a source of subjective sentiment knowledge, as in (6), where this knowledge is added to the previously described features (16), (17). Besides, many lexicon-based approaches take into account linguistic context (18), which involves mechanisms such as shifting the sentiment valence (19) with techniques like intensification (*very bad*) and negation (*not happy*). Nevertheless, lexicon-based approaches have many drawbacks: the necessity of labeled data that is consistent and reliable (18), the existence of expressions that vary

[1] http://googleresearch.blogspot.com.es/2016/05/announcing-syntaxnet-worlds-most.html

significantly among different domains, and the fact that lexicons can not be automatically translated for multilingual use. These disadvantages make hard to maintain a domain independent sentiment lexicon (20).

Due to the numerous research that has been done, many other features and learning algorithms have been tried. Like other supervised machine learning applications, the key aspect for sentiment analysis is the engineering of a set of effective features. Some of the many examples enumerated in (2) are:

> *Sentiment words and phrases.* Sentiment words are words that are used to express either a positive or negative sentiment. Most sentiment words are adjectives and adverbs (e.g., *good*, *bad*, *terrible*), but nouns (e.g., *rubbish*) and verbs (e.g., *hate*, *love*) can also contain sentiment information. Apart from this, there are also some idioms that express sentiment, e.g., *can't make an omelette without breaking eggs.*
>
> *Sentiment shifters and intensifiers.* These can be words that totally change the sentiment orientation, e.g., form positive to negative and vice versa. For example, "*I don't like this food*" is negative, and the sentiment shifter is *don't.* As for sentiment intensifiers, some examples are *very*, *many* and *few.*

An interesting example where many hand-crafted features are extracted in order to obtain a good performance in sentiment analysis is (21), where the domain is the Twitter realm. The features are: word n-grams, with n from 1 to 4; character n-grams, with n ranging from 3 to 5; number of words appearing in upper case; Part Of Speech tagging; number of hashtags; several scores based sentiment lexicons; characteristics about the punctuation, e.g., number of exclamation marks; presence of absence of emoticons, e.g., ':)'; number of elongated words and the appearance of negations words. As it can be seen, this set of feature is fairly complex, and is totally domain-dependent.

In general, extracting non-simple features from the text and figuring out which features are relevant or not, as well as selecting a classification algorithm, are fundamental questions in machine learning driven methods (22), (23), (24). Traditional approaches rely on manual feature engineering which is time consuming.

Deep techniques

Continuous representations of words as vectors has proven to be an effective technique in many NLP tasks, including sentiment analysis (25). One of the most relevant algorithms in

the field of deep learning for NLP are the DBOW and Skip-gram models, generally called Word2Vec models (10). These algorithms compute continuous vector representations of words form very large datasets. The extracted word vectors contain a huge amount of syntactic and semantic regularities present in the language, expressed as relation offsets in the vector space (26). These word-level embeddings are encoded by column vectors in an embedding matrix $W \in \mathbb{R}^{d \times |V|}$, where $|V|$ is the size of the vocabulary. Each column $W_i \in \mathbb{R}^d$ corresponds to the word embeddings vector of the i-th word in the vocabulary. The transformation of a word w into its word embedding vector r_w is made by using the matrix-vector product:

$$r_w = W v_w$$

where v_w is an one-hot vector of size $|V|$ which has value index at w and zero in the rest. The matrix W components are parameters to be learned, and the dimension of the word vectors d is a hyper-parameter to be chosen. The knowledge contained in the vector representations can result very effective when performing the task of Sentiment Analysis (27).

In deep learning for Sentiment Analysis (SA), an interesting approach is to augment the knowledge contained in the embedding vectors with other sources of information. This added information can be sentiment specific word embedding as in (25), or as in a similar work, a concatenation of manually crafted features and these sentiment specific word embeddings (28). Another approach that incorporates new information to the embeddings is described in (29), in which Deep Learning is used to extract sentiment features in conjunction with semantic features. (30) describe an approach where distant supervised data is used to refine the parameters of the neural network from the unsupervised neural language model. Also, a collaborative filtering algorithm can be used, as is detailed in (31), where the authors add sentiment information from a small fraction of the data. In the line of adding sentiment information, in (32) is portrayed how a sentiment Recursive Neural Network can be used parallel to another neural network architecture. In general, there is a growing tendency which tries to incorporate additional information to the word embeddings created by deep learning networks. An interesting work is the one described in (33), where both sentiment-driven and standard embeddings are used in conjunction with a variety of pooling functions, in order to extract the target-oriented sentiment of Twitter comments.

In deep learning, word are predominantly represented as vectors through the *word2vec* technique (10). The two main models that support this tool are Skip-gram and CBOW (Continuous Bag Of Words). While the Skip-gram model predicts surrounding words $w(t-$

$2), w(t-1), w(t+1), w(t+2)$ given a word $w(t)$, the CBOW model predicts a word $w(t)$ given the surroundings (Figure 2.2). In both models, the surrounding words are past and future words.

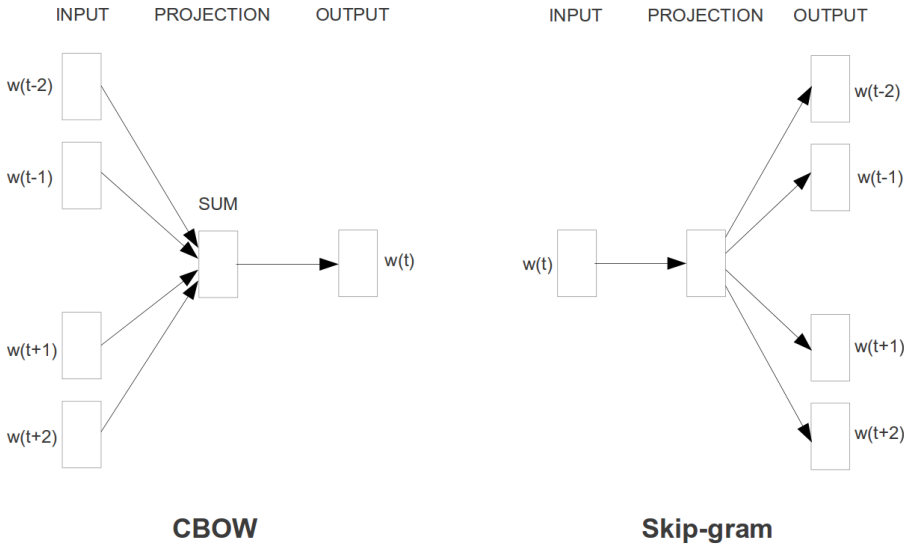


Figure 2.2: CBOW and Skip-gram architectures.

In the case of Skip-gram, the learning objective function can be expressed as the maximization of the average log probability (34):

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

where c is the size of the training context, or window. Larger c values can result in higher accuracy, but at the expense of training time. For the sake of completion, the basic formulation of Skip-gram defines $p(w_{t+j}|w_t)$ using the softmax function:

$$p(w_O|w_I) = \frac{\exp(v_{w_O}^T v_{w_I})}{\sum_{w=1}^{|V|}\exp(v'_w v_{w_I})}$$

where v_w and v'_w are the 'input' and 'output' vector representations of w. Each word is associated with an input and output vector, that is averaged to obtain the final word vector. This last formulation is impractical due to cost of computing $\bigtriangledown p(w_O|w_I)$ is proportional to $|V|$, which is often very large (around 10^5). In practice, more complex formulation is used, based on the hierarchical softmax.

As an extension of the Skip-gram model, the GloVe (Global Vector for Word Representation) model has been presented (35). This efficiently leverages statistical information, taking advantage in a better manner of this concurrence information that the Skip-gram

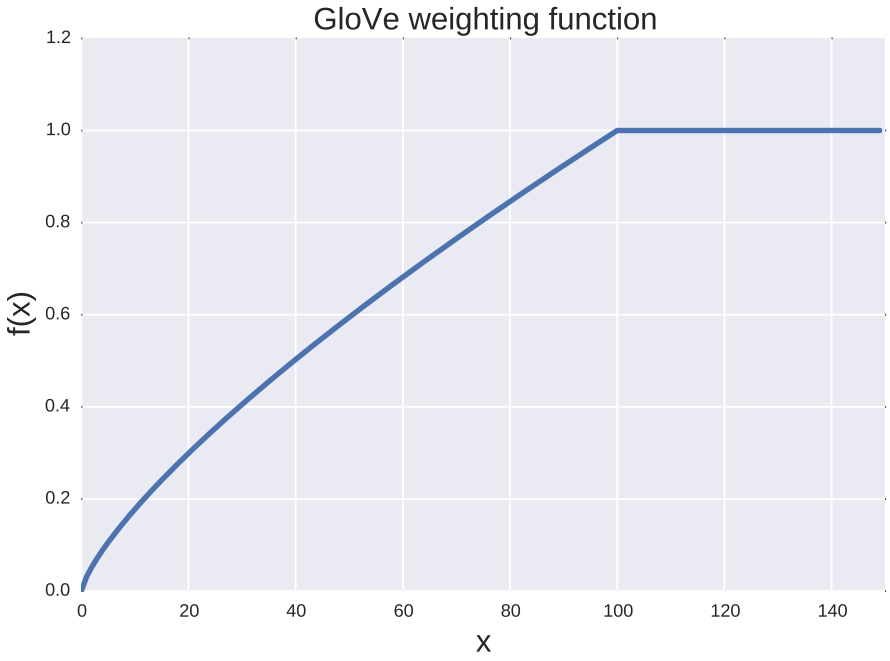


Figure 2.3: GloVe weighting function.

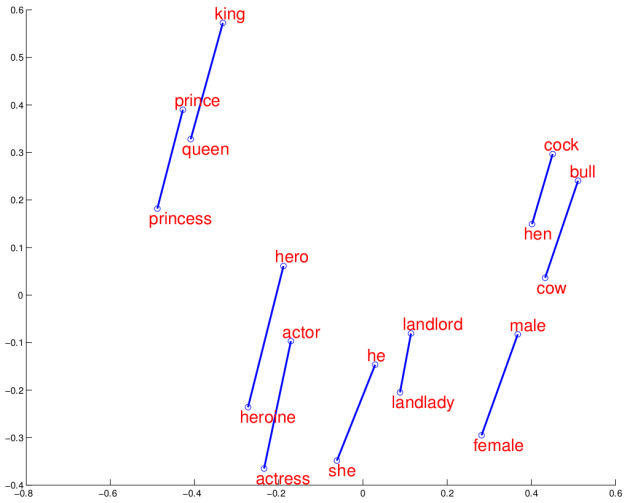
model. In short, this model introduces a weighting function to the original Skip-gram formulation that allows reducing the noise introduced by low occurrence tokens, augmenting the weight with the occurrence ratio. The function formulation is as follows:

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & \text{if } x \leq x_{max} \\ 1 & otherwise \end{cases}$$

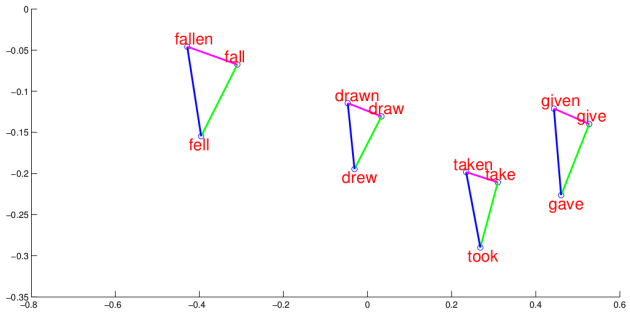
with α and x_{max} being hyper-parameters. Figure 2.3 shows a typical realization of this function, with $\alpha = 3/4$ and $x_{max} = 100$.

A very interesting characteristic of these word vector representations is that the word vectors can be combined using vector addition operations. These operations manifest the information that the vectors contain. For example, computing the operation $w^{king} - w^{man}$ is approximately equal to operating $w^{woman} - w^{queen}$. Observing these results, many authors conclude (34) that the offsets obtained with addition (and subtraction) operations represent the underlying concept. In the case of the example, the offset would represent the concept "kingly".

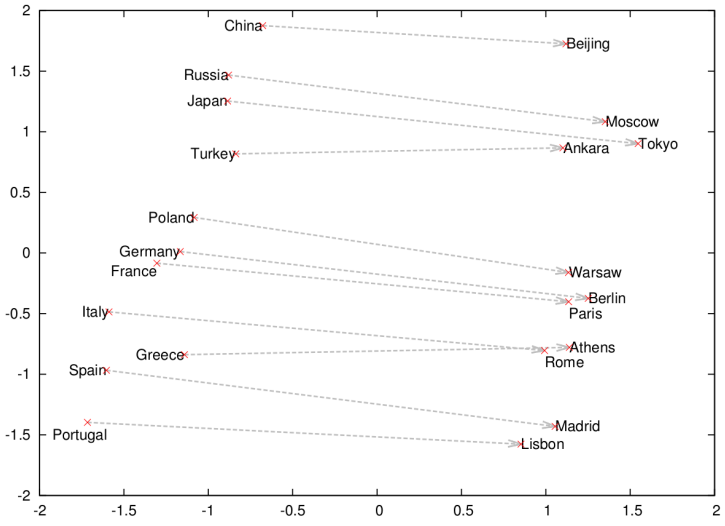
Another complex information is usually embedded into the continuous word vector repre-



(a)



(b)



(c)

Figure 2.4: 2D projections of the regularities between word vectors. (a) Male-female; (b) verb tenses; (c) country and capital.

sentations, such as verbal tenses ($w^{fall} - w^{fallen} \backsimeq w^{draw} - w^{drawn}$) and even country-capital relationships ($w^{japan} - w^{tokyo} \backsimeq w^{france} - w^{paris}$). This last type of offset is not syntactic, but mostly semantic. That is, these representations contain both types of information.

Moreover, it is possible to visualize the word vectors by projecting their values to a 2D plane. Although much information contained in the vectors is lost, the general structure is often visible. Any dimension reduction technique can be used. Normally, Principal Component Analysis (PCA), an Singular Value Decomposition (SVD) based algorithm, is used (9).

Alternatively, another newer visualization technique can be used: t-SNE (t-Distributed Stochastic Neighbor Embedding) (36). This technique is well-suited for the visualization of high-dimensional datasets. In some applications, t-SNE is better at preserving certain not linear characteristics of the data.

Nevertheless, it is worth mentioning that, in dimension reduction techniques, the resulting reduced dimensions do not have any direct meaning. For example, in PCA these dimensions are the ones that retain the most variance, and consequently information, of all the transformed ones.

It is worth mentioning that in the literature and in this work there is a distinction of two types of word vectors: generic and affect word vectors. Generic vectors, often referred to as pre-trained vectors, are word representations that have not been trained with an specific purpose or task. For example, training Skip-gram model on a Wikipedia dataset will result on pre-trained vectors, as this training has not been done with a specific task as objective. On the contrary, affect word vectors are those that have been trained for capturing *sentiment infomation*. This can be achieved by the use of neural networks that have in their objective function a metric of how well the produced vectors represent sentiment information. In this work, both types of word vectors are used.

Ensemble methods

In the field of ensemble methods, the main idea is to combine a set of models (component classifiers) in order to obtain a more accurate and reliable model in comparison with what a single model can achieve. In general, an ensemble model can be understood as

$$y_i = f(d_{1i}, d_{2i}, ..., d_{Li})$$

where $d_j i$ is the prediction of the j-th of L component classifiers in the i-th example of

the dataset, $f(\cdot)$ the ensemble function, and y_i the prediction of the ensemble.

The methods used to model combination are many, and a categorization is presented in (37). This classification is based on two main dimensions: how predictions are combined (Rule based and Meta learning), and how the learning process is done (Concurrent and Sequential).

Regarding the first dimension, on the one hand, in *rule based* approaches predictions from component classifiers are treated by a fixed rule, such as majority voting, with the aim of averaging their predictive performance. On the other hand, *meta learning* techniques use predictions from component classifiers as features for a meta-learning model.

In general, the fixed rule model combines multiple classifiers by computing their predictions d with a combination function $f(\cdot)$. The simplest way to combine multiple classifiers is the *voting* scheme, which corresponds to taking a linear combination of the learners (9):

$$y_i = \sum_{j=1}^{L} w_j d_{ji}$$

where $w_j \geq 0, \sum_j w_j = 1$, y_i is the prediction for the i-th example and d_{ji} is the prediction of the j-th classifier for the i-th example. This technique is also known as linear opinion pool. In the simplest case, all learner are given equal weight, and the result would be equivalent to taking an average. Nevertheless, computing a weighted averaged is one the of many possibilities of combination rules. An extensive set of fixed rules is showed in Table 2.1.

Meta-learning or *stacked generalization* models extends fixed rule models in that the output of the component classifiers is combined into another system, $f(\cdot|\Phi)$, which is yet another learner, and its parameters Φ are trained (9). The output is given as $y = f(d_1, d_2, ..., d_L|\Phi)$. An important fact is that the meta-learner should not be trained on the training data used for the component classifiers because then, these classifiers may memorize the data set. In fact, the meta learner should see the learner make errors. Therefore, the meta learner should be trained on training data unused in the process of training the component classifiers. In this models, $f(\cdot)$ has no restrictions, and can be a non linear function. Figure 2.5 shows the general architecture of an ensemble model.

As explained in (38), rule based ensembles can be quite effective in the task of sentiment classification. Besides, these fixed ensemble models can be added with extra subjective knowledge, as illustrated in (39), where a POS-based rule based ensemble model is proposed, obtaining quite good results in the task of cross-domain sentiment classification. Another rule-based ensemble approach is the one in (40), where a fixed rule based on the geometric

Rule	**Combination function** $f(\cdot)$
Sum	$y_i = \frac{1}{L}\sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji},\ w_j \geq 0,\ \sum_j w_j = 1$
Geometric mean	$y_i = \prod_j d_{ji}^{\alpha_k},\ 0 \leq \alpha_k \leq 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

Table 2.1: Fixed rules and their combination functions.

mean is used. Also, in (38) a meta-classifier ensemble model is evaluated, obtaining performance improvements as well. An adaptive meta-learning model is described in (41), which offers a relatively low adaptation effort to new domains.

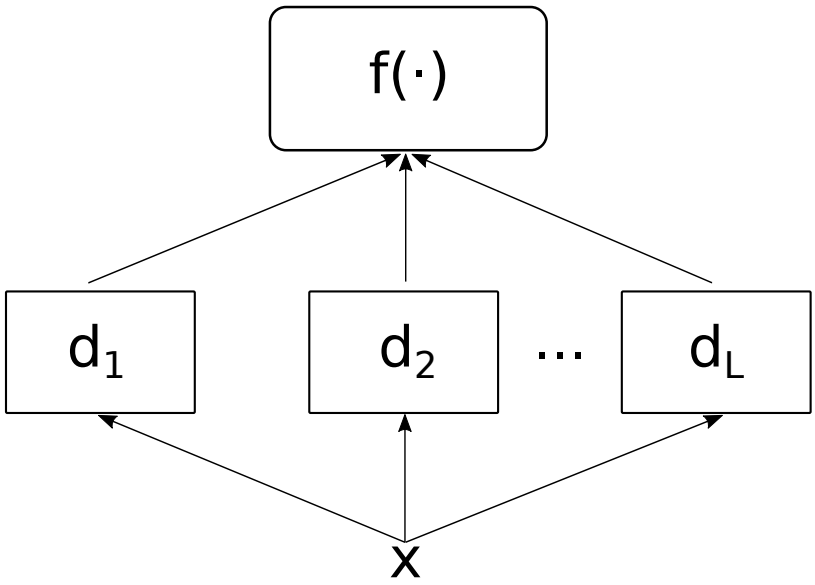


Figure 2.5: Ensemble architecture, where $f(\cdot)$ can be a fixed rule or a meta learner algorithm.

As for the second dimension, *concurrent* models divide the original dataset into several subsets from which multiple classifiers learn in a parallel fashion, creating a classifier composite. On the contrary, *sequential* approaches do not divide the dataset but there is an interaction between the learning steps, taking advantage from previous iterations of the

learning process to improve the quality of the global classifier.

The most popular technique that processes the samples concurrently is bagging (37). An example of bagging performance in the sentiment analysis task can be found in (42), where bagging and other classifications algorithms are used to show that sentiment and stock value are closely related. Bagging consists on building several instances of a black-box estimator on random subsets of the original training set, and then aggregate their individual predictions to form a final prediction. These methods are usually used as a way of reducing the variance of a complex classifier (e.g., a decision tree), by introducing a randomization factor and then combining the result. As they provide a way to reduce overfitting, bagging methods work best with strong and complex models. A *weak learner* has error probability less that 1/2, which makes it better than random guessing on a binary class problem, while a *strong learner* has a arbitrarily small error probability.

Regarding both concurrent and sequential approaches, a work by (43) shows that bagging and boosting (a sequential method) techniques can be very useful when dealing with large and noisy data. Another study (44) shows a comparison between bagging and boosting on a standard opinion mining task.

As opposed to bagging, boosting works by fitting a sequence of weak learners on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote to produce the final prediction. In boosting, the next learner in the chain is trained with the errors of the previous learner. In order to achieve this, the data modifications at each boosting iteration consist on applying weights w_1, w_2, ..., w_N to each of the training examples. Initially, those weights are set to $w_i = 1/N$, and so the first learner trains on the original data. For each successive iteration, the weights are modified and the learning algorithm is applied again to the weighted data. In this way, the incorrectly classified examples are weighted higher, and the correctly classified ones are given a lower weight.

Nevertheless, these works also show that ensemble techniques not always improve the performance in the sentiment analysis task, and there is not a global criteria to select a certain ensemble technique.

Ensemble taxonomy

This section presents the proposed taxonomy for ensemble techniques applied to Sentiment Analysis in both surface and Deep domains. This classification intends to summarize the

	S	**S+G**	**G**	**G+A**	**A**	**S+G+A**
No ensemble	(3)(4) (5), (12) (13), (14) (6),(18) (20), (16) (17)	(29), (31)	(7) (45) M_G	(30)	(25)	-
Classifier ensemble	(38), (39), (40), (41), (37), (42), (43),(44)	CEM_{SG}				CEM_{SGA}
Feature ensemble	(22), (23) (11)	M_{SG}, (28)		M_{SA} (32), (33)		M_{SGA}

Table 2.2: Proposed taxonomy for ensemble of surface and Deep features. S represents surface features, G and A stand for generic word vectors and affect word vectors, respectively. The combination of the features and/or word vectors is indicated with '+'. We consider the combination *No ensemble*/*S+G+A* not possible in the terms of this taxonomy.

work found in the literature as well as to compare these models with the ones we propose. Also, with this, we respond to the first question raised in Sec. 1.1 regarding how the different combination techniques can be classified.

The taxonomy can be expressed as combination of two different dimensions. On the one hand, considering which features (i) are used in the model, that can be either surface hand-crafted features, generic automatic word vectors, or affect word vectors specifically trained for the sentiment analysis task. On the other hand, attending to how the different model resources (ii) are combined: using no ensemble method at all, utilizing a classifier ensemble, or taking advantage of a feature ensemble (where the features are combined and fed to a learning algorithm). Table 2.2 shows a representation of this taxonomy, where the two dimensions appear as rows and columns. We have classified all the reviewed work in this paper using the proposed taxonomy, obtaining a visual layout of the techniques used in relation with ensembling methods with surface and deep features. The layout of the axis of the different ensemble approaches is shown next:

- *No ensemble.* This category includes analyzers based on surface features and not using any ensemble technique. Instead, it is considered that the models that fit in this category make use only of one type of feature.
- *Classifier ensemble.* Here are the tradition approaches that are based on ensemble techniques (2.3). This category considers that the combination is made between classifiers, hence the name classifier ensemble.
- *Feature ensemble.* This category contains the approaches that make use of surface feature combination techniques. Examples of this can be combining BOW with other kind of features, such as POS features, but also word vectors.

As for the axis that represents which features are used, there are several options:

- *S.* Surface features, such as BOW, POS or sentiment lexicons (2.1).
- *G.* Generic word vectors, such the ones obtained through the training process of the Skip-gram model (2.2).
- *A.* Affect word vectors that were specifically trained for embedding sentiment information.
- *Combinations.* This axis also takes into account the combination of different types of features: $S+G$ (surface features combined with generic word vectors), $G+A$ (generics word vectors with affect embbedings), and $S+G+A$ (all three types of features

combined in the same model). The combination $S+A$ is not represented, as well as the blank spaces in the taxonomy because of, to the extent of our knowledge, such techniques has not been studied, and so they represent work that can be addressed in the future.

CHAPTER 3

Sentiment Analysis Models

This master thesis explores the combination of surface and deep features. In order to successfully combine all these different types of information, it is necessary to take advantage of some aggregation methods. In this Chapter, the proposed combining models are presented. Firstly, the more straightforward approaches that ensemble several classifiers trained with different features. After this, the ensemble of features models are presented. These approaches combine different types of features -surface, generic and affect words- in order to augment the knowledge provided to the classifier.

This Chapter presents the sentiment analysis models we propose for sentiment analysis. These models have been validated in the Twitter domain (Sec. 4.1). First we describe the developed deep learning based analyzer used as baseline for the rest of the paper, and after this we detail several proposed ensemble models. These models are: ensemble of classifiers and ensemble of features. Regarding the ensemble of classifiers, we tackle two main approaches in further experiments: fixed rule and meta-learning models.

As for the ensemble of features, many combinations are studied: surface and generic embeddings, generic and affect embeddings, and the global combination of these three types.

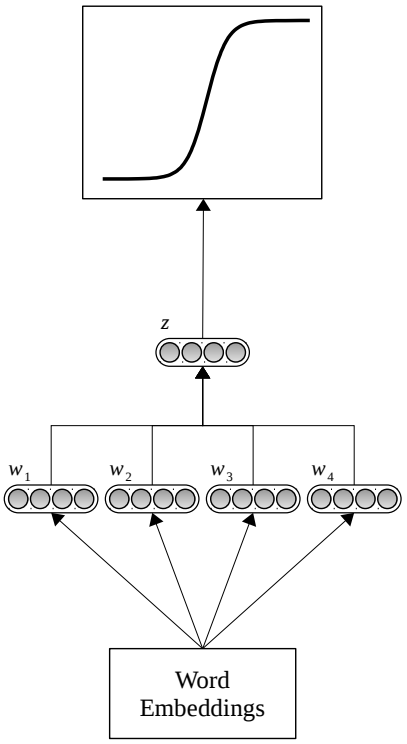


Figure 3.1: Schematic representation of the deep learning baseline model, M_G. In this figure, the word embeddings are combined into a single vector through one convolutional function, and the fed to a linear algorithm, which determines the polarity of the document.

Deep Learning classifier (M_G)

Generic word vectors, also denoted as pre-trained word vectors, can be captured by word embeddings techniques such as *word2vec* (10) and *GloVe* (35). Generic vectors are extracted in an unsupervised manner, not being trained for a specific task. These word vectors contain semantic and syntactic information, but do not enclose any specific sentiment information. Nevertheless, with the intention of exploiting the information contained in these generic word vectors, we have developed a sentiment analyzer model based on deep word embedding techniques for feature extraction, in order to compare it to other approaches in the task of Sentiment Analysis. This approach makes use of word embeddings and a linear algorithm.

The model composites a vector that represents the whole tweet, obtained through the *min*, *average* and *max* convolutional functions. These functions may be combined through the concatenation of its resulting vectors. Generally, let $w_1, w_2, ..., w_N$ be the N words of the tweet, and

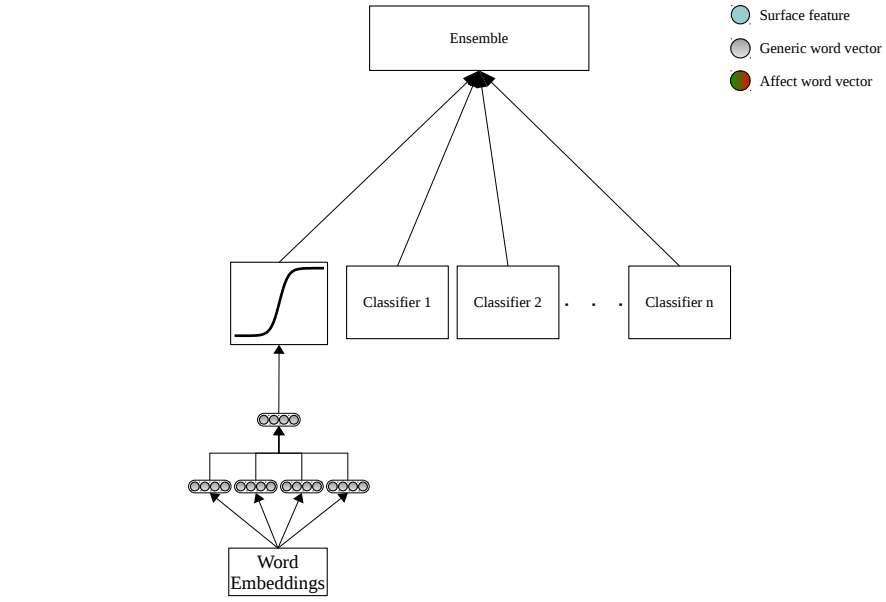
$$z(w_1, ..., w_N) = [f_{max}(w_1, ..., w_N), f_{min}(w_1, ..., w_N), f_{avg}(w_1, ..., w_N)]$$

the vector representing the whole tweet. The combination of n of these functions produces a vector of nd dimensions, being d the word vectors original dimension. For example, when only using the *average* function for representing the tweet, then $n = 1$ and z has a dimension of d.

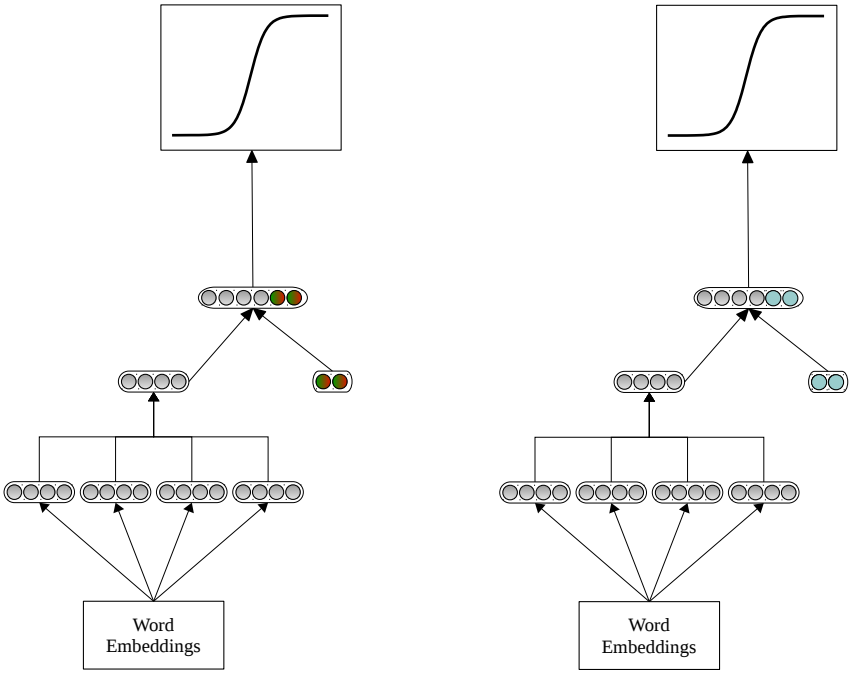
A diagram of this model is showed in Figure 3.1. We refer to this model as M_G, with the G standing for generic word vectors.

Ensemble of classifiers (CEM)

Our objective is to combine the information from surface and deep features. The most straightforward method is to make the combination at the classification level. In this way, we propose an ensemble model which combines classifiers trained with deep and surface features. We denote surface features as the ones extracted using hand-crafted methods or sentiment lexicons, as they rely on the surface of the text. In this way, knowledge from the two sets of features is combined, and this composition has more information than its base components. This model combines several base classifiers which make predictions from the same input data. These predictions can be subsequently used as new data for extracting a single prediction of sentiment polarity. Accordingly, this ensemble model is



(a) **CEM**$_{\mathbf{SG}}$



(b) **M**$_{\mathbf{SG}}$ (c) **M**$_{\mathbf{GA}}$

Figure 3.2: Diagram of how the different classifiers and features are combines in the $\mathrm{CEM_{SG}}$, $\mathrm{M_{SG}}$ and $\mathrm{M_{GA}}$ models.

aimed to improve the sentiment classification performance that each base classifier can achieve individually, obtaining better performance. The combination technique can be any treatment of the base classifiers predictions that outputs a sentiment polarity. Also, any number of base classifiers can be combined into this ensemble model. A schematic diagram of this proposal is illustrated in Figure 3.2.

We denote this model as *CEM*, which stands for Classifier Ensemble Model. The subscript indicates the types of features its base classifiers have been trained with, like in CEM_{SG}, where the ensemble combines classifiers trained with surface features and generic word vectors.

Next, the two ensemble techniques used in this ensemble model in further experiments are described.

Fixed rule model

This model seeks to combine the predictions from different classifiers using a simple fixed rule. Consequently, this ensemble does not need to learn from examples. The rules used in this approach can be any fixed rule used in ensembling models. In this work the rule used for the experiments' ensemble is the voting rule by majority. This rule counts the predictions of component classifiers and assigns the input to the class with most component predictions. In case of a match, a fixed class can be selected as the predicted by the model.

Meta classifier model

In the meta-classifier technique, the outputs of the component classifiers are treated as features for a meta-learning model. This approach has the advantage that can learn, so adapt, to different situations. As for the selection of the learning algorithm of this approach, there is not an indicative of which one should be used. In this work, we select the Random Forest algorithm, as it can achieve high performance metrics in sentiment analysis (46).

Ensemble of features (M_{SG} and M_{GA})

This model is proposed with the aim of combining several types of features into a unified feature set and, consequently, combine the information these features give. In this way, a learning model that learns from this unified set could achieve better performance scores that one that learns from a feature subset.

In this sense, we can distinguish two types of ensembles of features: ensemble of features that combines both surface and Deep Learning features, and an ensemble of features that were completely extracted using Deep Learning techniques. In this last type of ensemble of features, we stress the relevance of combining generic word vectors and affect word vectors. We refer to affect vectors as the result from training a set of pre-trained word vectors for an specific task, which in this case it would be SA.

The combination of features and/or word vectors is made by concatenation of the vectors into a unified feature vector, that is then fed to a linear algorithm. For example, for curating the feature vectors for the $\mathrm{M_{SG}}$ model, the $z_G \in \mathbb{R}^{d_G}$ word vector (Sec. 3.1) is used in combination with the vector $z_S \in \mathbb{R}^{d_S}$ resulting from the surface features extraction from the text:

$$z_{SG} = [z_G, z_S]$$

and $z_{SG} \in \mathbb{R}^{d_G+d_S}$, where d_G is the dimension of the generic word vectors, and d_S the dimension of the feature vector extracted trough surface techniques.

In the same manner, the model $\mathrm{M_{GA}}$ uses the concatenation of the generic word vectors and the affect word vectors. As it is described in Sec. 3.1, the affect vector representing a tweet is composed using three convolutional functions, *min*, *max* and *average*. And so, the composition of the z_{GA} vector is made as:

$$z_{GA} = [z_G(w_{G1}, ..., w_{GN}), z_A(w_{A1}, ..., w_{AN})] \in \mathbb{R}^{d_G+d_A}$$

with z_G being the same generic vector as in Sec. 3.1 that represents a single tweet, and z_A the affect vector that represents, also, a single tweet composed by N words,

$$z_G(w_{G1}, ..., w_{GN}) = [f_{max}(w_{G1}, ..., w_{GN}), f_{min}(w_{G1}, ..., w_{GN}), f_{avg}(w_{G1}, ..., w_{GN})]$$

$$z_A(w_{A1}, ..., w_{AN}) = [f_{max}(w_{A1}, ..., w_{GN}), f_{min}(w_{A1}, ..., w_{AN}), f_{avg}(w_{A1}, ..., w_{AN})]$$

where w_{Gi} is the generic word vector of the i-th word in a tweet, and w_{Ai} is the affect word vector of the i-th word in a tweet.

As for the $\mathrm{M^{SGA}}$, the model is the same, combining the three types of features:

$$z_{SGA} = [z_S, z_G, z_A] \in \mathbb{R}^{d_G+d_S+d_A}$$

We address to this first model type as M_{SG}, as it combines surface features and generic word vectors. In the same way, the second is referred as M_{GA}, combining both generic and affect word vectors. Lastly, an architecture where surface features, generic word vectors and affect word vectors are combined is denoted by M_{SGA}. A diagram representing two instances of the model is shown in Figure 3.2.

CHAPTER 4

Evaluation

In this chapter, the evaluation of the proposed models is made. The evaluation is based on several experiments made on Twitter datasets that aim to characterize the sentiment classification performance of this master thesis different models. Also, this evaluation makes use of the Friedman statistical test, using it to characterize the different models in a rank. As the statistical results imply, the proposed models significantly improve the baseline. Besides, two models stand out in this improvement.

Evaluation Setup

This section describes the experiments conducted in order to answer the questions formulated in the introduction (Sec. 1.1). These experiments are aimed to evaluate the performance between the deep learning baseline we have developed (M_G) and the proposed ensemble models. Each performance experiment is made with three different datasets, widely used by the community of Sentiment Analysis. All the performance metrics have been obtained using K-fold cross validation, with folds of 10.

Some proposed experiments are also aimed to characterize the sentiment analysis performance for each individual classifier, including our own approach, so later it can be compared to the ensemble models. For this purpose, we have collected several sentiment analyzers for composing a classifier ensemble.

As for the sentiment analysis of the natural language, it is conducted at the tweet level, so it is not necessary to split the input data into sentences. The classifiers label each comment as either positive or negative.

Datasets

The datasets used for testing are *SemEval 2013*, *SemEval 2014*(47), *Vader*(48) and *STS-Gold* (49). These datasets are uniquely used for testing purposes, so they are not used for training any algorithm. Also, we use the *Sentiment 140* (50) dataset for developing our deep learning baseline, M_G. These datasets are described next, and some statistics are summarized in Table 4.1.

Also, the distribution of the number of words for each dataset is showed through and histogram for both aggregated and separated in sentiment polarities. In these graphics, the mean value in each case is represented by a vertical line. When showing the distributions by sentiment polarity, the green symbolizes positive, while the red represents the negative sentiment. These graphics help to gain insight about the dataset contents, and the possible differentiation by sentiment polarity.

The *SemEval 2013* test corpus is composed if 8,987 English comments extracted from Twitter on a range of topics: several entities, products and events. Similarly, we have also use the *SemEval 2014* test dataset. In both *SemEval* datasets, the data is not public but must be downloaded from the web first. As some users have already deleted their comments online, we have not been able to recover the original datasets, but subsets of it. Besides, as the development dataset contains only binary targets (positive and negative), we have

made an alignment processing of the *SemEval* datasets, filtering other polarity values.

Figure 4.1 shows the distribution of the number of words in each tweet of the SemeEval2013 and 2014 datasets. It is worth mentioning that the two *SemEval* datasets are very similar, with both number of words distribution alike. Also, the mean values of these distributions are very close, nearly 20 words per tweet. The differences between positive and negative tweets in the distributions do not seem relevant.

The *Vader* dataset contains 4,200 tweet-like messages, originally inspired by real Twitter comments. A subset of these messages is specifically designed to test some syntactical and grammatical features that appear in the natural language.

For this dataset, word distribution is showed in Figure 4.2. Vader dataset has a different distribution that the SemEval datasets. The mean value of the word count is lower, around 15 word per comment. Also, there are some differences in the distributions for positive and negative polarities. Nevertheless, the mean value is practically equal for both sentiment polarities.

The last dataset used for testing is the *STS-Gold* dataset for Twitter, which has been collected as a complement for Twitter sentiment analysis evaluations processes (49).

For this last test dataset, the word count distributions are showed in Figure 4.3. It is interesting to note that the aggregated distribution of the STS and the Vader datasets are very similar, unlike the sentiment-dependent distributions. Also, their mean values are close to 15 words per comment/tweet. While the word count distributions help to describe a dataset, they do not give any insight regarding how well a certain classifier performs when analyzing the sentiment of that dataset. Proof of this can be found when attending how similar the distributions between the Vader and STS datasets are, and how different is the performance of the same models (with no additional training) on these datasets.

As for the training data of our baseline model, the selected dataset is the *Sentiment 140* dataset, containing 1,600,000 Twitter messages extracted using a distant supervision approach (50). The abundance of data in this dataset is very beneficial to our deep learning approach, as it requires large quantities of data to extract a fairly good model, as pointed out by (10).

The word count histograms of this dataset are gathered in Figure 4.4. The Sentiment140 dataset has a unique word count distribution, very different to that of the rest of datasets. One would expect the distributions of the SemEval and Sentiment140 datasets to be alike, as these datasets are completely composed of user-generated tweets. Nevertheless, their distributions are very different. While the SemEval datasets have a bell-shaped distributions,

Dataset	Positive	Negative	Total
SemEval2013	2,315	861	3,176
SemEval2014	2,509	932	3,441
Vader	2,901	1,299	4,200
STS-Gold	632	1,402	2,034
Sentiment140	800,000	800,000	1,600,000

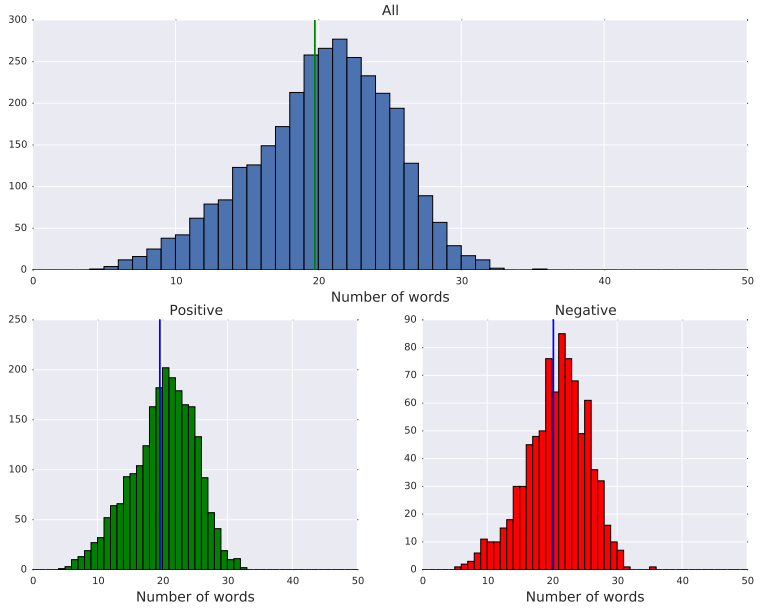
Table 4.1: Statistics of the SemEval2014/2014, Vader, STS-Gold and Sentiment140 datasets.

the Sentiment140 dataset has not that shape. This fact could be explained attending at how the Sentiment140 dataset was collected: through distant supervision (50). Only the tweets that contain a *emoji*, such as :), ;) and :(were included in this dataset. In fact, the distant supervision comes from using this emoji information to label each tweet as either positive or negative.

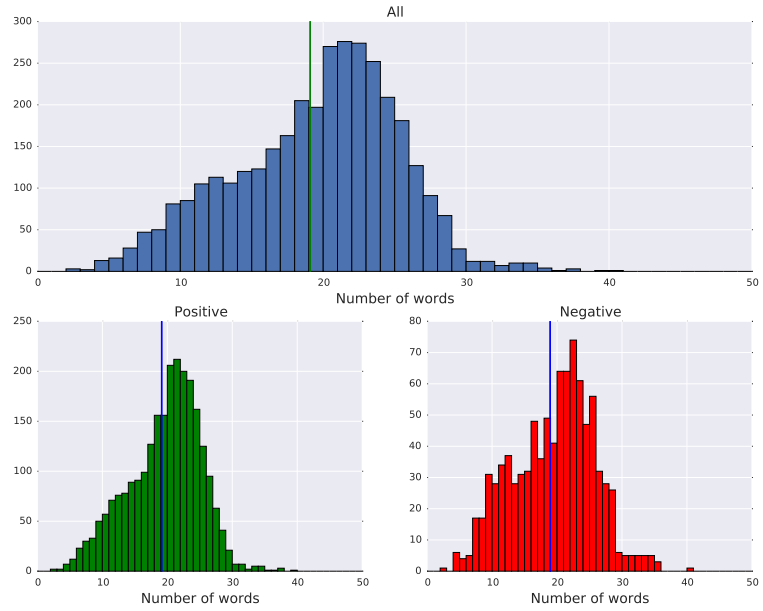
Evaluation metrics

In sentiment analysis, as in many information retrieval tasks, the *accuracy score* is not always a good indicator of the performance of a system. We can define the accuracy score as the number of correct predictions made divided by the total number of predictions made. This score is interesting when the test dataset is balanced, that is, there is approximately the same number of examples in all the classes (e.g., in a binary sentiment prediction dataset, the number of positive examples is similar to the number of negative examples). Nevertheless, when the dataset is not balanced, the accuracy score can be misleading. For example, is a dataset that has the 90% of its examples labeled as positive, a dummy system that predicts only positive sentiment would obtain a accuracy score of 90 %, and the accuracy score would not be a good indicator of this system's performance.

For this reason, the metric used in this work is the macro averaged F1-Score, a metric that can acknowledge for imbalanced datasets. This metric is the harmonic mean of two values, *precision* and *recall*. For defining these two values, the analysis task must be understood. Figure 4.5 shows a typical sentiment analysis scenario. For example, when accounting for documents with positive sentiment:



(a)



(b)

Figure 4.1: Distribution of the number of words by sentiment polarity and aggregated for SemEval2013 (a) and SemEval2014 (b).

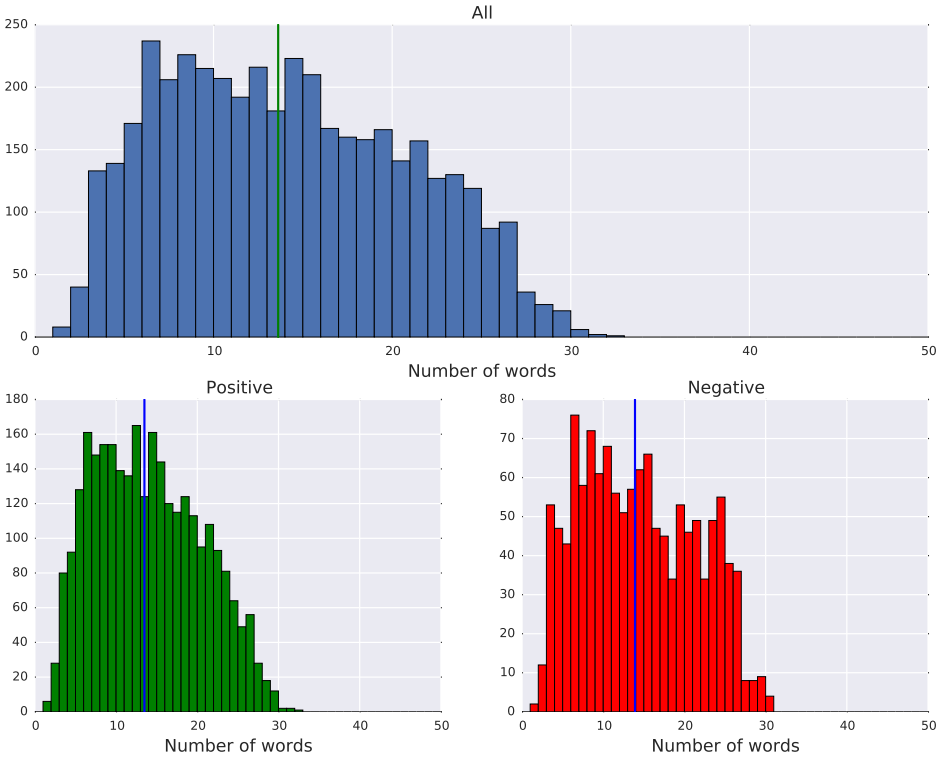


Figure 4.2: Number of words distribution by sentiment polarity and aggregated for Vader dataset.

True positives are the documents correctly classified as positive.

True Negatives are the document with negative sentiment that have been correctly classified as negative.

Type I errors (or false positives) are negative documents that have been incorrectly classified as positives.

Type II errors (or false negatives) are positive documents that hava been incorrectly classified as negative.

Once this scenario has been defined, we can define precision and recall as:

Precision. Number of correct positive results divided by the number of all positive results.

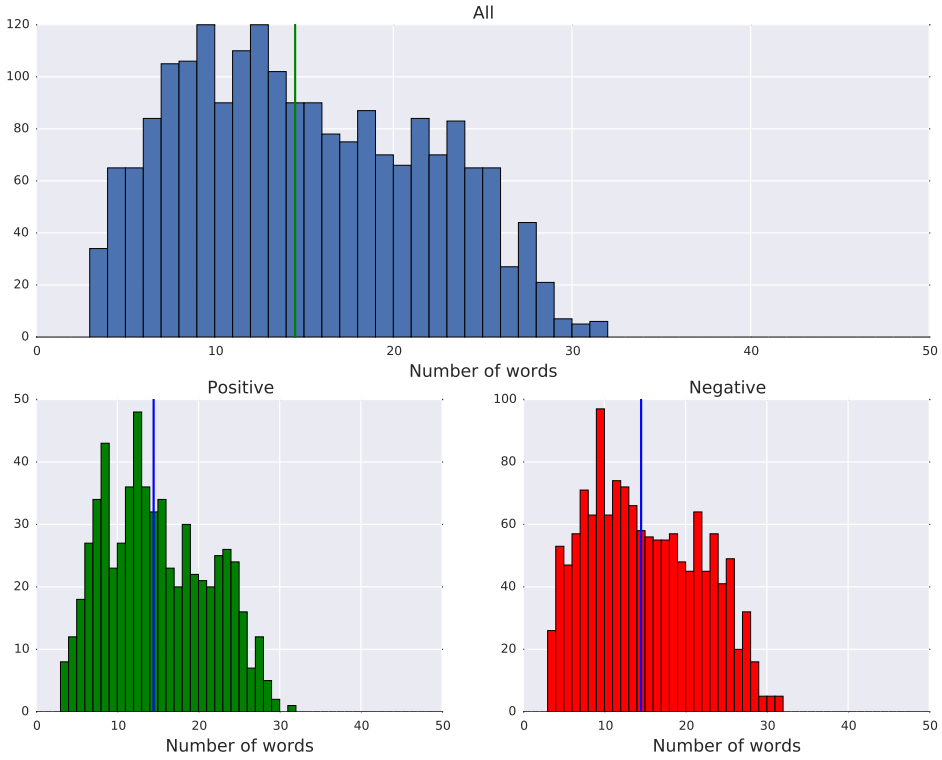


Figure 4.3: Number of words distribution by sentiment polarity and aggregated for STS dataset.

$$P = \frac{TP}{TP + FP}$$

Recall. Number of correct positive results divided by the number of positive results that should have been predicted.

$$R = \frac{TP}{TP + FN}$$

The F1-Score can be interpreted as a weighted average of these two values, reachinh its best score at 1, and worst at 0. The expression for the F-Score is:

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} = 2\frac{P \cdot R}{P + R}$$

being P and R the precision and recall, respectively. In this way, the F-Score combines

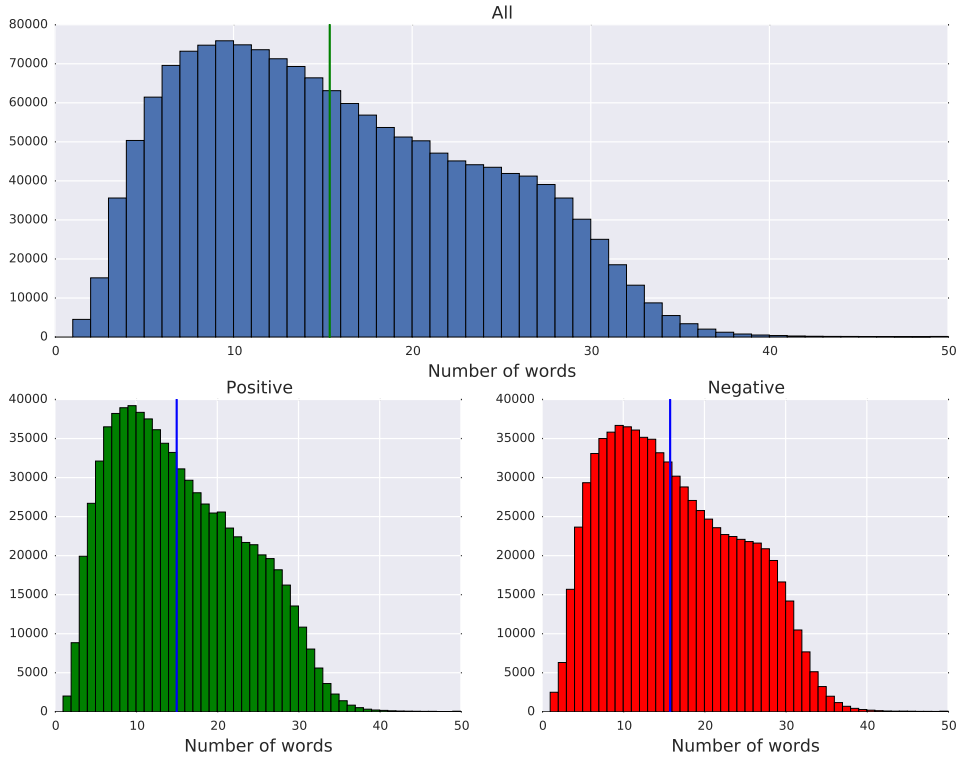


Figure 4.4: Number of words distribution by sentiment polarity and aggregated for the Sentiment140 dataset.

the precision and recall values into a single value, giving information of the performance of a classification system in a numerical piece of information.

Baseline training

For the implementation of the M_G model we have utilized the popular Word Embeddings model known as *word2vec* (10). We use the implementation of the Word2Vec algorithm of the gensim library (51).

The dimension of the vectors generated by word2vec is 500. We used a recursive feature selection algorithm implemented in scikit-learn (52) to discover the optimal dimension for the downstream application of Sentiment Analysis once the pre-trained word vectors were obtained. The dimension which achieves the best performance is 500[1]. For higher

[1]The result of the feature selection algorithm is 497 dimensions, but for simplicity of the implementation we choose all 500 dimensions, as this does not have an impact on the classification performance.

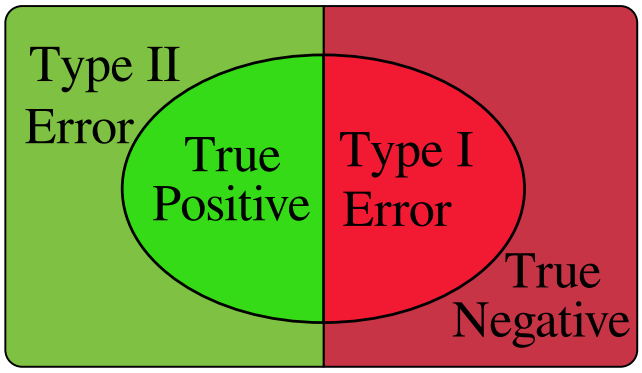


Figure 4.5: Sentiment analysis errors types scenario.

dimensions, the performance improvement is not relevant.

As for the word2vec model training, we use 1,280,000 tweets randomly selected from the Sentiment 140 dataset. Once this model is extracted, we feed a linear regression model (implementation from scikit-learn) with the vectors of each comment and the labels from the original dataset. We split the sentiment 140 dataset using the ratio 80/20 for training and development set, respectively.

With respect to the convolutional functions, we have conducted an effectiveness test of the *max*, *average* and *min* functions on the development set. The results are showed in Table 4.2. As can be seen, the *avg* function is very close to the performance of the complete set of functions *max*, *avg* and *min*. Consequently, we select the *avg* function as the one used for further experiments, as it provides very good results compared to the rest, and it also reduces the computational complexity of the experimentation.

Regarding the preprocessing of the natural language, we tokenized the input data and removed punctuation, excepting the most common ('.,!?'). We also transformed URLs, numbers and usernames (@username) into especial characters, with the aim to normalize the data. The preprocessing is applied to all the texts before generating the word vectors.

Ensemble of classifiers

In order to improve the performance of the Deep Learning baseline, we have built an ensemble composed of this analyzer and six different sentiment classifiers. Following, a list of each of these classifiers is shown:

- sentiment140 (50)
- Stanford CoreNLP , (53)

Convolutional function	F-Score
max	74.82
avg	77.53
min	74.99
max + avg	77.63
max + min	76.7
avg + min	77.70
max + avg + min	77.73

Table 4.2: Effectiveness of the convolutional functions on the development dataset.

- Sentiment WSD (54)
- Vivekn (55)
- pattern.en (56)
- TextBlob Sentiment Classifier (57)

We have built ensemble classifiers using two combining techniques in the CEM_{SG} model: a rule based method and a meta learning approach, both using the predictions of the classifiers composing the ensemble as features for the next step. For the meta-learning approach, we use the implementation of scikit-learn of the Random Forest algorithm. For this algorithm we have used 500 as a number of estimators, as in the experiments the value of this parameter did not make any relevant change to the classification performance in the range from 50 to 1,000. This exploration can be seen in Figure 4.6.

Ensemble of features

For the implementation of the surfaces features, we used the following, based on the work by (21): Wordnet-Affect lexicon values (58), number of exclamation, interrogation and hashtags marks '!?#', numer of positive, neutral and negative words, number of words that are all in caps and number of words that have been elongated 'goooooood'.

	SemEval2013	**SemEval2014**	**Vader**	**STS-Gold**
sentiment 140	78.92	60.67	78.76	75.07
CoreNLP	46.95	42.95	60.19	59.68
WSD	76.18	75.35	77.75	75.35
vivekn	72.14	59.97	63.54	69.61
pattern	82.50	71.86	85.98	82.86
TextBlob	82.51	71.92	85.71	68.27
M_G	85.34	86.16	87.71	83.43
CEM_{SG}^{Vo}	87.78	84.16	87.92	83.52
CEM_{SG}^{MeL}	**87.87**	87.63	88.85	84.56
M_{SG}	86.36	87.03	88.07	84.73
M_{GA}	87.54	88.05	88.89	85.27
M_{SGA}	86.26	86.94	88.89	85.26
CEM_{SGA}^{Vo}	86.26	85.90	**89.52**	**87.08**
CEM_{SGA}^{MeL}	86.97	**88.07**	89.48	85.59

Table 4.3: Macro averaged F-Score of all the sentiment classifiers. Bold metrics are the best in the test dataset. Besides, the last row shows the result of the Friedman test, in bold the two best models. (Legend) **sent140**: sentiment140, **CoreNLP**: Stanford CoreNLP, **WSD**: Sentiment WSD, **TB**: TextBlob.

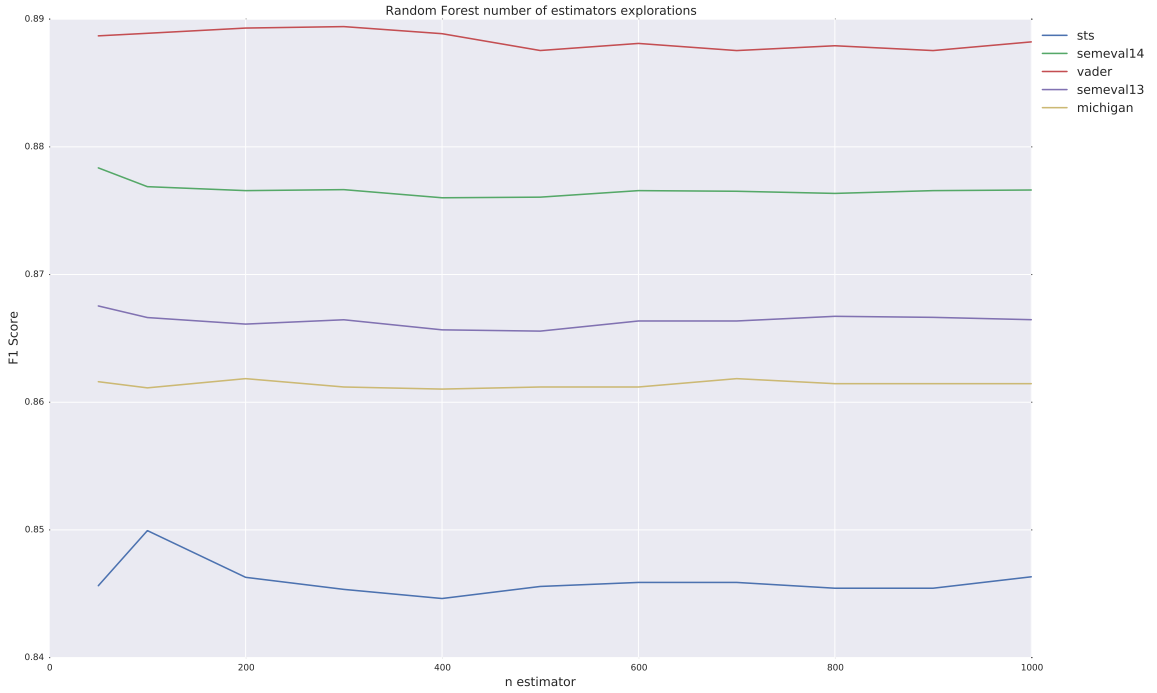


Figure 4.6: Exploration of the number of estimators hyper-parameter of the Random Forest algorithm.

As for the M_{GA} model, we use the word vectors obtained by (25). More specifically, we use the vectors extracted using the $SSWE_r$ neural model. These vectors have been extracted for learning sentiment-specific word vectors but not general semantic information, so we use them as affect word vectors.

Analysis

The experiments we conducted show the sentiment classification performance of each base classifier separately (including our deep learning baseline) on each of the three test datasets, as well as the metrics for the ensemble of classifiers and several ensembles of features. In this section, the experimental results are shown and discussed. The results of all the experiments are gathered in Table 4.3.

Performance of the base classifiers

The better F-score performance is achieved by TextBlob in SemEval2013, by the WSD classifier with an important difference over TextBlob in SemEval2014; while pattern.en has

a slightly better performance than the rest in the Vader dataset and has also the better performance in the STS-Gold dataset by far. Furthermore, the classifier with the lower performance is CoreNLP in all four test sets. The CoreNLP low performance in these experiments can be due to the fact that this classifier is not well adapted to the short reviews and comments such the ones found in the Twitter domain.

The mean accuracy performance for the base classifiers is 73.02, 63.79, 75.32 and 71.81 % in the SemEval2013, SemEval2014, Vader and STS-Gold datasets respectively.

Classifiers and features classifiers performance

CEM models gather the predictions from M_G baseline and the other six base classifiers whose classification performance has been analyzed. The voting and meta-learning techniques are used as ensembling techniques. It can be seen in the Table 4.3 that these ensemble models surpass the baseline, as well as all the other base classifiers. In fact, the best performance for all the test sets are achieved by these CEM classifiers.

As for the feature ensemble models, they also push the performance further than the baseline. The M_{GA} ensemble is very close to the best metrics in both SemEval datasets, and is relatively close to the best performance in the Vader dataset. Also, it seems that M_{SGA} is suffering overfitting, as the combination of all three types of features decreases the performance when comparing to M_{GA} model.

Moreover, as an another observation, it can be seen that the better improvements are achieved by CEM^{SGA}_{Vo} and CEM^{SG}_{MeL} models, with 3.65 and 2.53 % of performance gain in STS-Gold and SemeEval2013 datasets respectively.

Statistical analysis

In order to compare the different proposed models in this work, a statistical test has been applied on the experimental results. Concretely, the Friedman test with the corresponding Bonferroni-Dun post-hoc test, that are described by (59). These test are specially oriented to the comparison of several classifiers on multiple data sets.

Friedman test is based on the rank of each algorithm in each dataset, with the best performing algorithm gets the rank of 1, the second best gets rank 2, etc. Ties are resolved by the average of the ranks. r^i_j is the rank of the j-th of the k algorithms and on the i-th of N datasets. Friedman test uses the comparison of average ranks $R_j = \frac{1}{N}\sum_i r^j_i$, and states that under the null-hypothesis (all the algorithms are equal so their ranks R_j are equal) the

Friedman statistic, with $k-1$ degrees of freedom, is:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left(\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right)$$

Nevertheless, (59) shows that there is a better statistic that is distributed according to the F-distribution, and has $k-1$ and $(k-1)(N-1)$ degrees of freedom:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$$

#classifiers	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$q_{0.05}$	1.960	2.241	2.394	2.498	2.576	2.638	2.690	2.724	2.773
$q_{0.1}$	1.645	1.960	2.128	2.241	2.326	2.394	2.450	2.498	2.539

Table 4.4: Critical values for the Bonferroni-Dunn test. The number of classifier includes the control classifier.

If the null-hypothesis of the Friedman test is rejected, post-hoc tests can be conducted. In this work we employ the Bonferroni-Dunn test, as it allows to compare the results of several algorithms to a baseline. In this case, all the proposed models are compared against $\mathrm{M_G}$. This test can be computed through comparing the critical difference (CD) with a series of critical values (q_α), summarized in Table 4.4 (59). The critical difference can be computed as:

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}}$$

$\mathbf{M_G}$	$\mathbf{CEM_{SG}^{Vo}}$	$\mathbf{CEM_{SG}^{MeL}}$	$\mathbf{M_{SG}}$	$\mathbf{M_{GA}}$	$\mathbf{M_{SGA}}$	$\mathbf{CEM_{SGA}^{Vo}}$	$\mathbf{CEM_{SGA}^{MeL}}$
7.5	6	3.75	5	**2.875**	4.75	3.875	**2.25**

Table 4.5: Friedman average ranks R_j for all proposed studied models.

For the computation of both tests, the ranks have been obtained. The average ranks (R_j) are showed in Table 4.5. The α values is set to 0.1 for the following calculations. With those averages, the Friedman statistics:

$$\chi_F^2 = \frac{12\cdot 4}{8\cdot 9}\left((7.5^2+6^2+3.75^2+5^2+2.875^2+4.75^2+3.875^2+2.25^2) - \frac{8\cdot 9^2}{4}\right) = 13.48$$

$$F_F = \frac{3 \cdot 13.48}{4 \cdot 7 - 13.48} = 2.78$$

and the critical value $F(k-1,(k-1)(N-1)) = 2.02$. As $F_F > F(7,21)$ (59), the null-hypothesis is rejected and the post-hoc test can be conducted. According with this, the q_α in this case is 2.45, and the critical difference

$$CD = 2.45 \cdot \sqrt{\frac{8 \cdot 9}{6 \cdot 4}} = 4.24$$

Continuing with the Bonferroni-Dunn test, the difference between the average ranks of the baseline and the j-th model is compared to the CD and, if greater, we can conclude that the j-th algorithm performs significantly better than the baseline.

Friedman test has pointed the $\text{CEM}_{\text{SGA}}^{\text{MeL}}$ and the M_{GA} models as the two best classification models, followed by the $\text{CEM}_{\text{SG}}^{\text{MeL}}$ and $\text{CEM}_{\text{SGA}}^{\text{Vo}}$ models. Following, the Bonferroni-Dunn test finds that $\text{CEM}_{\text{SGA}}^{\text{MeL}}$ and M_{GA} models perform significantly better that the baseline. As for the rest of the models, it is not possible to reach a conclusion with the current data.

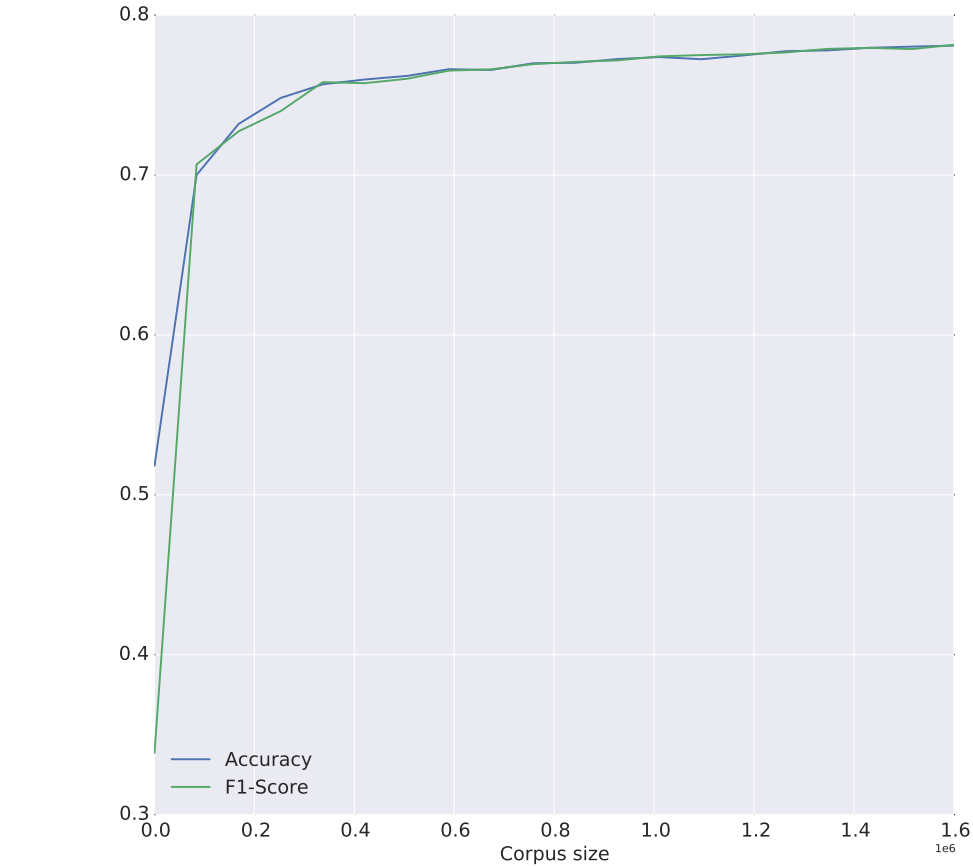
On top of this, an interesting result of these experiments is that the performance of the meta learning approach is higher that one of the fixed rule scheme. While the meta learning ensemble with all types of features (SGA) is significantly better than the baseline, the voting model is not. This could be caused by the learning capabilities of the meta-classifier technique, feature that the fixed ensemble methods like voting rule do not have.

Other experiments

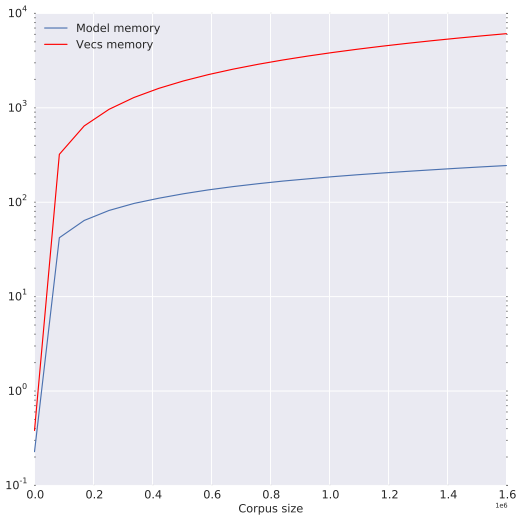
This section describes additional experiments made in this master thesis. These experiments were made in order to explore the utility of the word vectors generated by the *word2vec* word embedding tool (10) for sentiment analysis, as well as the scaling performance of the word2vec implementation used (51). Concretely, we delve into the capabilities of these embeddings to represent sentiment information while also retaining all the characteristics detailed in Sec. 2.2. In fact, the results of these explorations could be useful for the sentiment analysis.

Scaling Word2Vec

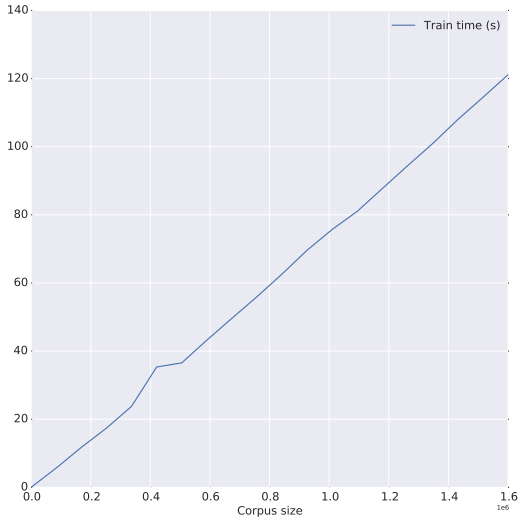
As described in (10), the Skip-gram model improves its quality, and also its utility for sentiment analysis, as the size of the training corpus augments. In this experiment we



(a)



(b)



(c)

Figure 4.7: Word2Vec scaling performance tests: (a) accuracy and f1 score in sentiment analysis, (b) memory occupied by corpus and model vectors -the vertical axis is in logarithm scale- and (c) training time in seconds.

explore this property by feeding the model with a corpus in a iterative process, increasing the training size. The corpus of the experiment is the Sentiment140 corpus. We use the whole 1,600,000 tweets. At each step several metrics are computed: the 10-fold accuracy and f1 score in the sentiment analysis task with the M_G model 3.1; memory occupied by the vectors computed from the dataset; memory occupied by the Skip-gram model; and time of training. The experiment results are shown in Figure 4.7.

Regarding the effect of the corpus size in the sentiment analysis performance, it can be seen that, indeed, the increase on the corpus size improves the performance. It is interesting to see that when the training corpus is small, the accuracy is much higher than the f1 score. Nevertheless, these two metrics reach the same value around the 100,000 examples in the training dataset, and then they do not diverge. Besides, around the 400,000 exaples in the trainig dataset, these two metrics start stabilize. In other words, the growth of the model performance slows down. That said, the model performance continues to grow, thought more slowly. In fact, the growth does not stop in all the experiment.

Respecting the memory used by both the model vectors and the corpus vectors, it can be seen that these last vectors consume between 1 and 2 orders of magnitude more that the model vectors. In reality, this result indicates that the number of words in the model vocabulary and the number of examples in the dataset are related by the same factor. As the vertical axis of the Figure 4.7b is in logarithmic scale, the vectors of both the model and the dataset grow linearly with the training corpus size.

Lastly, the timing profile of the training process in shown in Figure 4.7c. This result clearly indicates that the training time of this word embeddings model behaves linearly with the corpus size.

Sentiment seeding

It is well known that the word embeddings generated by models such as *word2vec* retain many semantic and syntactic knowledge on the language they are trained on, and this information can be even be aligned to perform language translations (26; 60). In an attempt to discover similar relations between the word vectors and their sentiment characteristics, we designed an experiment in which we search for similarities in the vectors with similar sentiment meaning.

In these word embeddings, one could expect two words with similar semantic meaning to be close on the d-dimensional space, such as *cat* and *dog*. In the same manner, two words with very different semantics should be more afar from each other, like *computer* and

cheese. In the same manner, we could expect these relations in a sentiment form: the word *good* could be close to the word *exceptional*, but farther away from the word *bad*.

In order to discover this relations, we have used a *seeding* technique, where two seeds were chosen to be the representatives of each sentiment polarity. So, we selected the word *excellent* for being the seed for the positive sentiment, and the word *horrible* as the negative sentiment seed. Once the word were selected, we extracted the 20 word vectors more close to each seed, obtaining 21 positive words vectors, and the same amount for the negative vectors. As for the model used for obtaining the word vectors, we used the public word2vec model trained on 100 billion word of the Google News dataset [2].

The metric which measures the closeness of two word vectors in the d-dimensional space, we used the *cosine similarity* metric, as outlined in (60). This metric can be written as

$$cos(\theta) = \frac{v \cdot w}{||v|| \cdot ||w||}$$

with the result varying from -1, meaning completely opposite, to 1, which means exactly the same. When this metric results in 0, it means orthogonality (or decorrelation).

As mentioned in Sec. 2.2, the word vectors can be visualized in a 2D plane, and still retain enough information regarding the original non-compressed vectors to draw conclusions from them. In this experiment, we have used the PCA algorithm to reduce the word vectors from $d = 300$ to $d' = 2$.

The result of the seeding and dimensionality reduction can be seen in Figure 4.8. It is immediately seen that these two clusters of words are arranged separately, attending to the positive or negative sentiment of each word. Once it has been verified that the word vectors retain some sentiment information, this 'sentiment clusters' could be used in sentiment analysis systems.

In fact, this technique has already been used in (61), where the highest consine similarity is used as a feature for a sentiment classifier.

Sentiment lexicon clusters

Continuing with the exploration analysis of the word embeddings capabilities showed in Sec. 4.3.2, another study was made on the word vectors. This experiment follows a different approach: using a sentiment lexicon in order to search for sentiment similarities between the words of this lexicon.

[2] https://code.google.com/archive/p/word2vec/

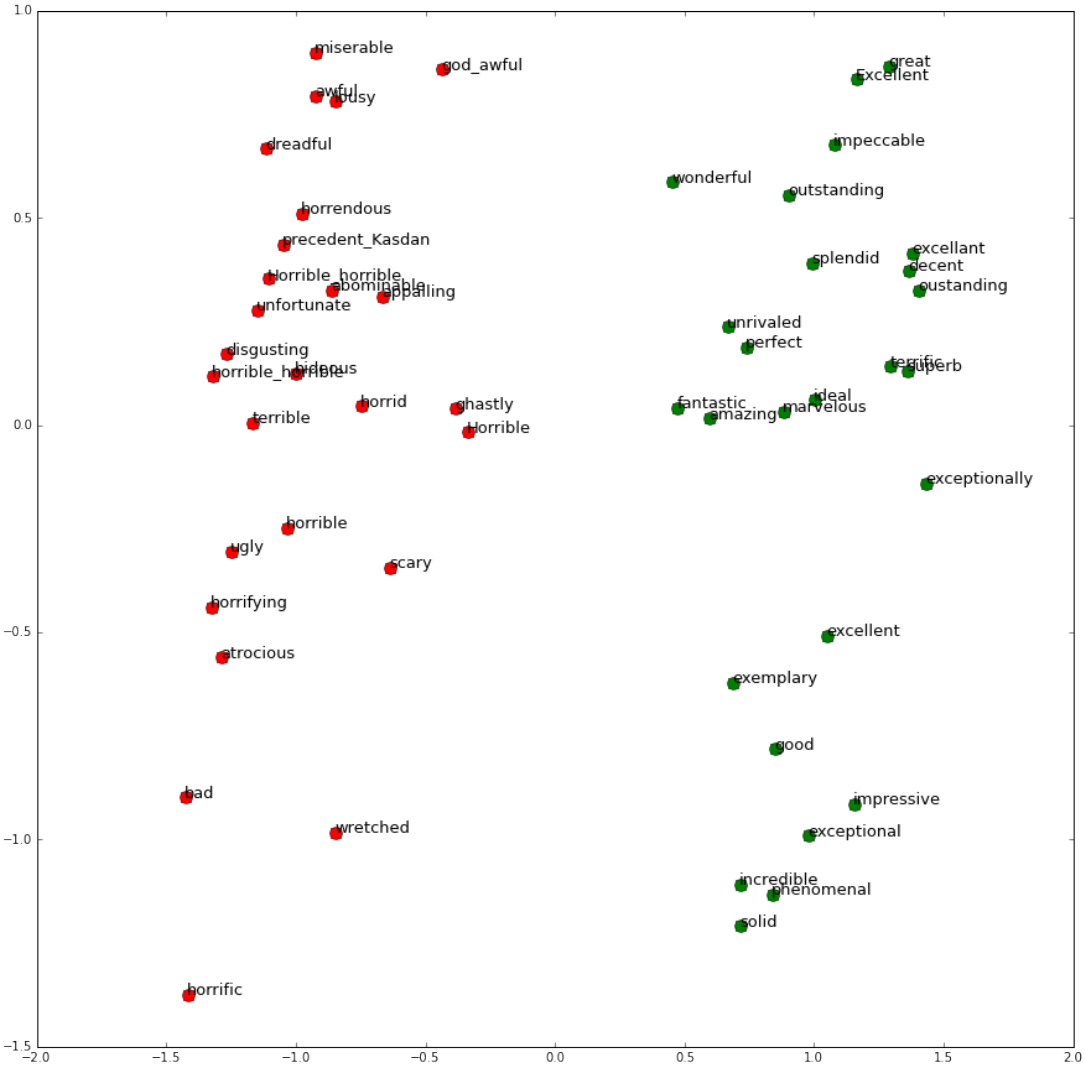


Figure 4.8: Sentiment seeds and their 20 more close neighbors in the d-dimensional space. In this plot, the seeds are *excellent* (positive in green) and *horrible* (negative in red).

Concretely, the approach taken in this experiment is, given a already generated word embeddings models, to extract all the words vectors of a lexicon, and to reduce their dimensionality to 2D using PCA. In this way, it can be rapidly checked whether the word embeddings have sentiment simialarities or not. For this experiment, we use the same word2vec model used in 4.3.2, and the sentiment lexicon described in (62). This lexicon contains 2006 positive words, and 4783 negative words.

The result of the experiment is shown in Figure 4.9. It can be observed that the word vectors arrange in two different clusters, corresponding to the two sentiment polarities. The result is not as clear as the seeding experiment (4.3.2), but the clusters can be easily detected.

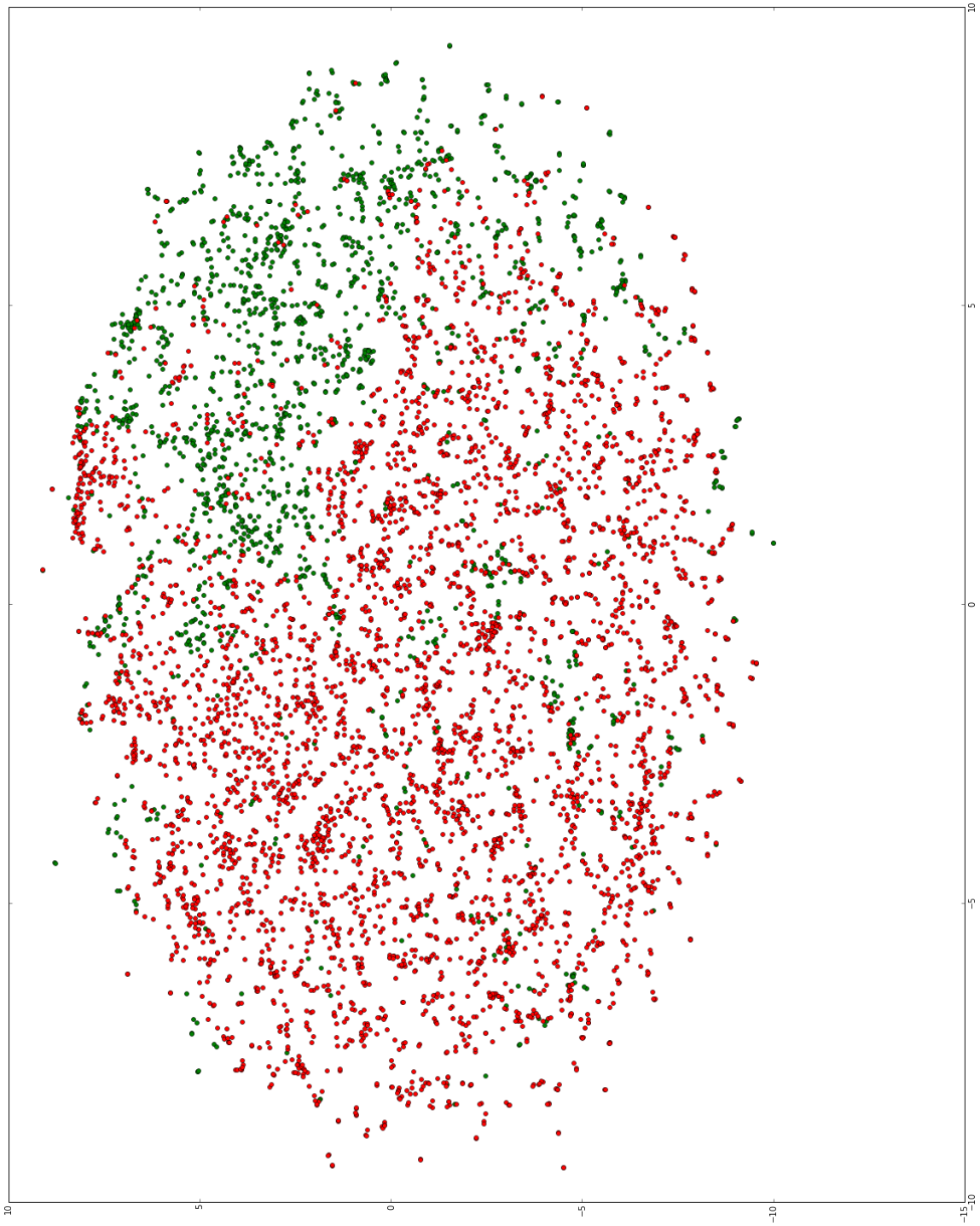


Figure 4.9: Lexicon word vectors represented in a two dimensional plane, reduced using PCA. Positive sentiment is showed green, while negative is red.

Still, this is an interesting result, as it shows that word vectors trained with no objective, in a unsupervised manner, retain sentiment information. Thus, we think that these characteristics can be used in sentiment analysis tasks as additional information.

Also, the quality of these word embeddings in the sentiment detection can surpass that of the lexicons, as pointed out in (25), where a quality experiment is carried out. In this work, specific sentiment word vectors (affect word vectors) had more quality that the pre-trained word embeddings, and than the sentiment lexicons.

In is worth mentioning, as made in Section 2.2, that the specific word embeddings are learned automatically, with hardly any human intervention, while the sentiment lexicons are manually curated in a work-intensive process.

CHAPTER 5

Case Study

This Chapter describes the case study oriented to aspect based sentiment analysis on the domain of Amazon reviews. The challenges raised by the change of domain and the highest granularity in the analysis pose new challenges, that must be addressed with additional techniques to the ones already presented. In this regard, in this Chapter we present a context detection algorithm that aims to detect the set of words that relate to a certain aspect. This algorithm is based on the extraction of the dependency tree. That is, the syntactical structure of a document. Moreover, this Chapter also presents the experimental results from the analysis in this alternative domain.

This chapter aims to describe selected use case. This study is not a thorough evaluation of the models proposed in Chapter 3 but a prototype, a measure of how well the proposed models perform in a sentiment analysis task of a different domain. Besides, this chapter also describes an algorithm used in ABSA tasks, which has been implemented in this master thesis.

The goal of this case study is to test the proposed models in a completely new domain, very different from the original one. While the proposed models are general, that is, they are not specially adapted to a certain domain, in Chapter 4 the evaluation is made only in the Twitter domain. In order to address this potential deficiency in the evaluation, in this chapter we take these models to a new domain, the review of products. This is an important step in the evaluation of the models, as the Twitter and the reviews domain are very different.

Regarding the sentiment analysis task, in this chapter we descend a level further in the classification of sentiment analysis (Sec. 1.2), and study the proposed techniques in the Aspect Based Sentiment Analysis realm. The proposed models in this master thesis have been designed for sentiment analysis at document and sentence level. In fact, in Chapter 4 we considered the tweets as a document, but they could also be treated as sentences, due to its length limitations (140 characters each tweet). But in this case study, we use the models differently, in the context of the aspect level.

Also, in this study, we combine the proposed models with a context detection algorithm. This algorithm detects, in a sentence review, the words that are related to an aspect. In this way, this context is then passed to a sentiment analysis model. With this, the sentiment analysis of an aspect is made only related to the words regarding that aspect.

Setup

This section describes the case study setup. First, the used datasets are described. After this, the context detector is described, including its use with sentiment analysis models.

Datasets

In relation to the domain of the analysis, we selected the reviews of phones. Concretely, the review of two Nokia mobile phones. The dataset used for this is the one found in (63). This dataset is a ABSA benchmark, as is annotated with positive and negative sentiment polarity.

there is much which has been said in other reviews about the features of this phone , it is a great ***phone*** *, mine worked without any problems right out of the box . but after several years of torture in the hands of* ***at&t customer service*** *i am delighted to drop them , and look forward to august 2004 when i will convert our other 3 family-phones from at&t to t-mobile !*

i have had the phone for 1 week , the ***signal quality*** *has been great in the detroit area (suburbs) and in my recent road trip between detroit and northern kentucky (cincinnati) i experienced perfect signal and reception along i-75 , far superior to at 38; t 's which does not work along several long stretches on that same route . but i spent hours setting up the stations (accepts about 13-14 , i believe) , though the* ***reception*** *is unpredictable .*

Figure 5.1: Extraction from the dataset of phone reviews. In bold, the aspects of the document and their polarity: green for positive, red for negative.

Figure 5.1 shows an extract of the phone reviews dataset. In these sentences, the aspects (in bold) can be seen. When applying sentiment analysis techniques to these documents, the contexts of the different aspects must not be mixed, as it would affect the sentiment analysis quality.

The phone dataset is divided in two datasets. The first dataset, *mobile phone 1*, is composed of reviews about certain products. In this case, the products are mobile phones and its accessories. As for the second dataset, it is a constructed dataset that ensures that each review combines both positive and negative sentiment. The statistics of the dataset are shown in Table 5.1. As it can be seen, the polarity labels are equally distributed in the two datasets.

Dataset	Positive	Negative	Total
Mobile phone 1	43	43	86
Mobile phone 2	62	62	124
Cell phone and accesories	148,657	24,343	173,000

Table 5.1: ABSA datasets statistics and collected Amazon dataset.

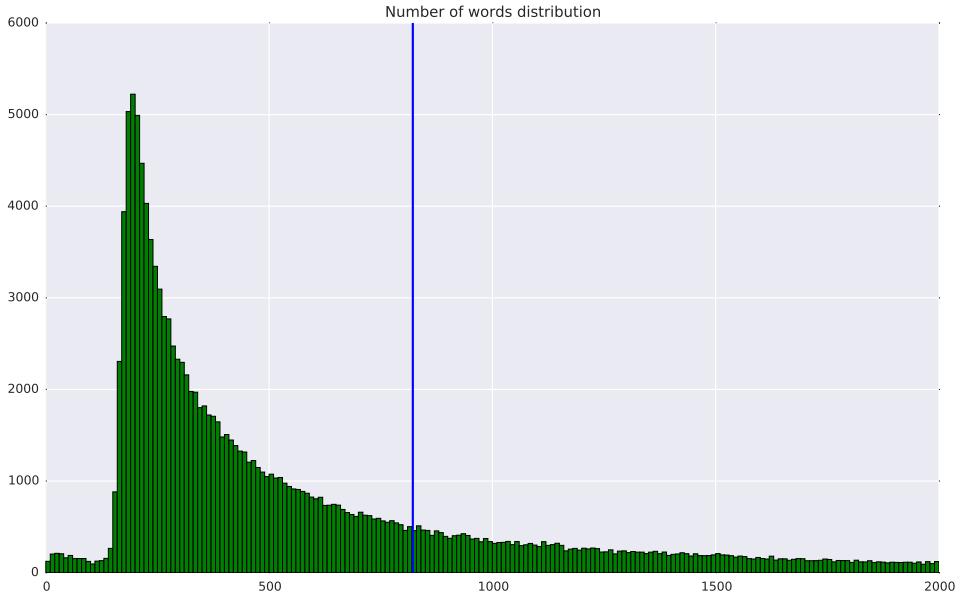
Added to this public datasets, we have collected a subset of the *Amazon product data* (64; 65). In concrete, we have extracted the review texts from all the reviews referring to cell phones and accesories. As for the polarity labeling, we have made use of the Amazon rating system, where each review has associated a value between 1 to 5, where 1 express the lowest satisfaction with the product, and 5 the highest. We have considered the reviews labeled with 4 or 5 stars as positive, and the ones with 1 or 2 to be negative. In Table 5.1 the statistics of this dataset are summarized.

Figure 5.3 shows the distribution of the number of words in the collected Amazon dataset. Also, the mean value is expressed by a vertical in this same Figure. Figure 5.2 shows this distribution separated by sentiment polarity. These distributions express that the majority of reviews has around 300 words. Also, there reviews that are longer (the tail of the distribution extends beyond 2,000 words). Moreover, there is a minority of reviews that have 200 or less words. As for the separation between sentiment distributions, there is not a relevant difference.

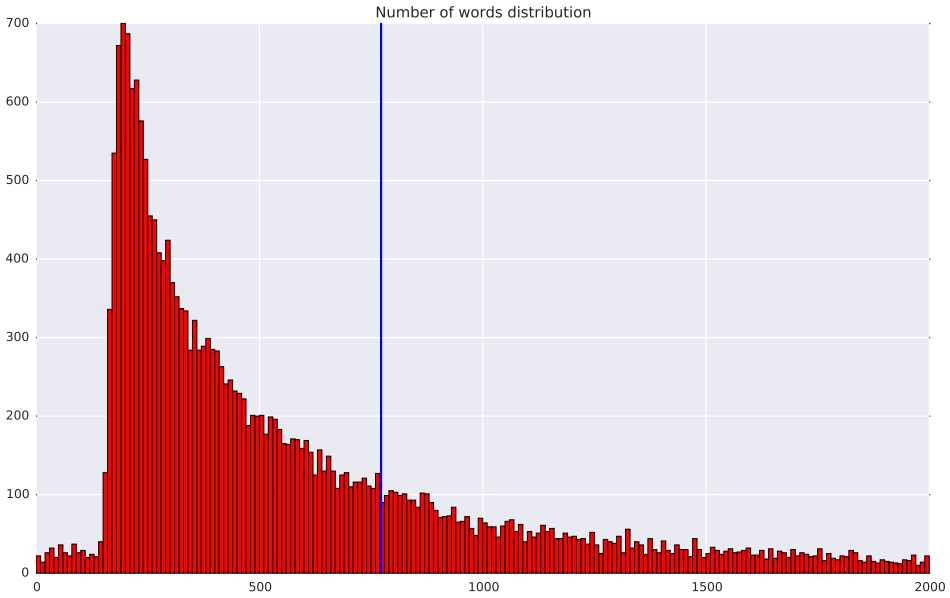
Sentiment Analysis system

In the sentiment analysis system, we can distinguish two main components: the context detector and the sentiment analysis analyzer. The context detection algorithm is in charge of detecting the context of the document aspects, which are then passed to the sentiment analyzer, that classifies the aspects' contexts.

In this case study, the aspect detection task is not addressed. The aspects are detected through the dataset. In this way, only the performance of the combination of the aspect context detection algorithm and the sentiment analyzer is measured.



(a)



(b)

Figure 5.2: Distribution of the number of words in the collected review dataset by sentiment polarities: positive (a), and negative (b).

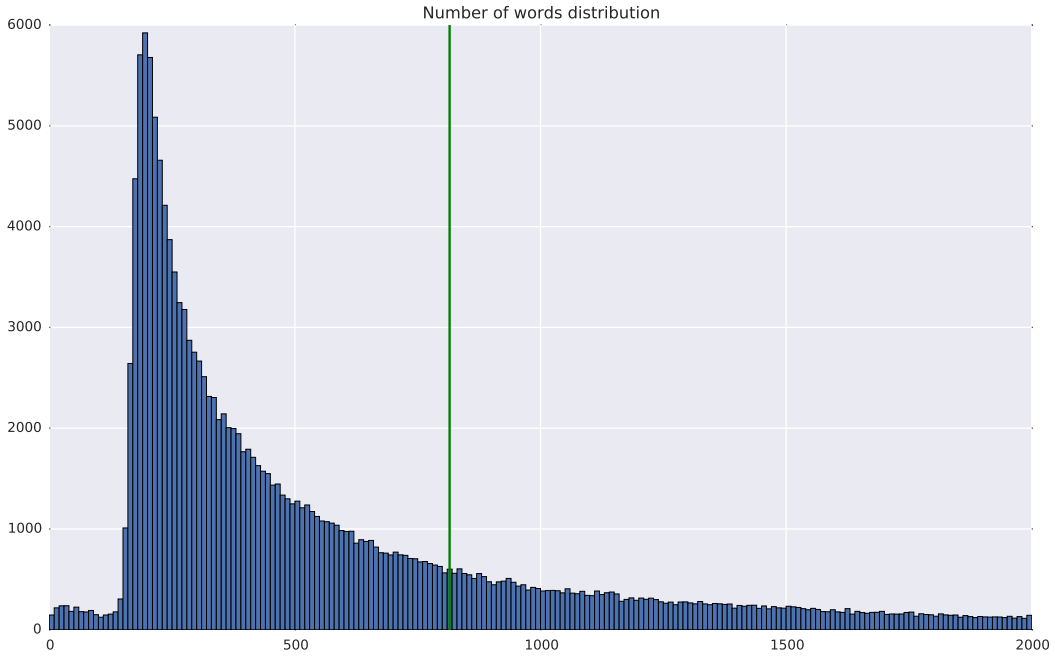


Figure 5.3: Number of words distribution in the collected review dataset.

Context Detection

This module is in charge of detecting the context of the aspects in the review documents. The context of an aspect is the set of words that refer an opinion to the target aspect. This capacity becomes very relevant when analyzing the sentiment of mixed opinions, where a set of sentences, or even one sentence express the opinion of different aspects, and potentially, these opinions are different. As an example, Figure 5.4 shows two mixed opinions in the same sentence. It can be seen that these opinions are totally opposed, and that they are referring to different aspects.

I like the looks of the phone, but the battery life is really bad.

Figure 5.4: Mixed opinions in a same sentence, one referring to phone design, and other to battery life.

An interesting solution to the opinion detection in the context of mixed sentiments is context detection. In this master thesis, we have implemented the algorithm proposed in

(63). This algorithm extracts the opinion words referring to the target feature using the dependency relations between words in a sentence. The dependency relations are those that relate two words, involving a syntactical meaning. An example of a dependency parsing is shown in Sec. 2.1.

Once the dependency tree has been extracted, it offers a measure of distance between words attending to number of arcs that link each word in the dependency graph, when going from one word to another. In other words, the distance $d(w_i, w_j)$ between two words w_i and w_j is the number of arcs, or edges, connecting them in the shortest path.

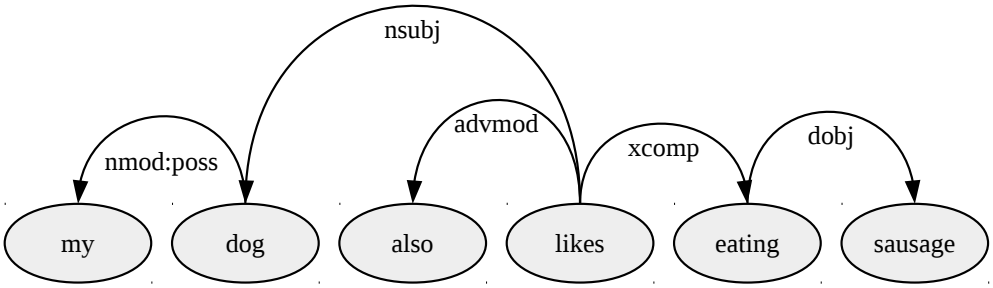


Figure 5.5: Dependency tree of a simple sentence.

Besides, the distance between two words w_i and w_j can be expressed as a sum of the distances of these words to an intermediate word, w_k.

$$d(w_i, w_j) = d(w_i, w_k) + d(w_k, w_j)$$

where w_k is a intermediate word in the dependency tree, not in the natural order of the sentence.

The intuition behind these distances is that more closely related words come together to express an opinion about an aspect (63). If there are n aspects of a product in a sentence, then the word that are more close (in terms of these dependency distances) to an aspect i will come together to express some opinion about it, rather than about another aspect j, to which they are not so close.

Besides, the dependency tree can be expressed more compactly as a *graph matrix* G. In this matrix, the rows and columns represent the words of the sentence, and its contents the distance to each other. The matrix obtained directly from parsing the dependency tree is not complete useful, as many relations are not specified. Continuing with the example in

Figure 5.5, the matrix associated to that tree is expressed as:

$$G = \begin{array}{c} \\ my \\ dog \\ also \\ likes \\ eating \\ sausage \end{array} \begin{array}{c} \begin{array}{cccccc} my & dog & also & likes & eating & sausage \end{array} \\ \begin{pmatrix} 0 & 1 & & & & \\ 1 & 0 & & 1 & & \\ & & 0 & 1 & & \\ & 1 & 1 & 0 & 1 & \\ & & & 1 & 0 & 1 \\ & & & & 1 & 0 \end{pmatrix} \end{array}$$

Note that G is a symmetric matrix, with the diagonal elements all zero. That is, $G_{i,j} = 0$ if $i = j$. Besides, as it is symmetric, $G_{i,j} = G_{j,i}$ for all indices i and j.

The rest of the elements of G, those that are no assigned a value when parsing the dependency tree, are considered to have a infinite value (∞). Nevertheless, in order to correctly express all the relations in the graph, the matrix must be changed, so that it expressed all the shortest paths. In this way, we select the Dijkstra's algorithm for computing the shortest paths in the dependency graph. After applying the Dijkstra's algorithm, the graph resulting is

$$G = \begin{array}{c} \\ my \\ dog \\ also \\ likes \\ eating \\ sausage \end{array} \begin{array}{c} \begin{array}{cccccc} my & dog & also & likes & eating & sausage \end{array} \\ \begin{pmatrix} 0 & 1 & 3 & 2 & 3 & 4 \\ 1 & 0 & 2 & 1 & 2 & 3 \\ 3 & 2 & 0 & 1 & 2 & 3 \\ 2 & 1 & 1 & 0 & 1 & 2 \\ 3 & 2 & 2 & 1 & 0 & 1 \\ 4 & 3 & 3 & 2 & 1 & 0 \end{pmatrix} \end{array}$$

After this, the distances in the graph can be obtained attending to the elements of G. For example, the distance between the words *dog* and *sausage* is expressed in the matrix as $G_{dog,sausage} = G_{2,6} = 3$.

At this step, we have a graph represented by a matrix G, and a set of aspects. The goal of this algorithm is not to detect, given a sentence, which word or words represent the aspects, and so in this discussion that information is supposed to be already given. We denote this set of aspects as A, and the graph of relations as R.

In general, as detailed in (63), there are n aspects where n is the dimension of A. The

algorithm for extracting the set of words $w_i \in S$ that express any opinion about the target aspect a_t proceeds as follows:

Algorithm 1 Dependency extraction algorithm

(i) Initialize n clusters $C_i \forall i = 1..n$

(ii) Make each $a_i \in A$ the clusterhead of C_i. The target aspect a_t is the clusterhead of C_t. Initially, each cluster only consists of the clusterhead.

(iii) Assign each word $w_j \in S$ to cluster C_k s.t.
$k = arg\ min_{i \in n} d(w_j, a_i)$

(iv) Merge any cluster C_i with C_t if $d(a_i, a_t) < \theta$ where θ is some threshold distance.

(v) The set of words $w_i \in C_t$ expresses the opinion regarding the target aspect a_t.

In other words, n clusters are initialized, each cluster C_i corresponding to each feature $a_i \in A$, being a_i the clusterhead of C_i. Then, each word $w_i \in S$ is assigned to the cluster whose clusterhead is closest to it. After this, any cluster is merged with the cluster whose clusterhead is the target aspect if the distance between their clusterheads is lower than a threshold θ. Finally, the set of words in the cluster C_t give the opinion about the aspect a_t.

For example, when detecting the contexts for the aspects in the sentence "*I like the looks of the phone and it is a great buy, but the battery life is really bad*"). In this sentence, the aspects could be *phone*, *buy* and *battery life*, and the target aspect is *phone*. In order to extract the contexts of these aspects, the dependency parsing must be done, obtaining a dependency tree. With the graph constructed from the tree, we compute the graph matrix G. Once the graph matrix is obtained, the next step is to apply the Algorithm 1. In this case the Algorithm detects the following clusters for all the aspects:

phone$\rightarrow$ *i like the looks of the phone and*

buy$\rightarrow$ *it is a great buy, but*

battery life$\rightarrow$ *the battery life is really bad*

As the target aspect is *phone*, the algorithm tries to merge the clusters with the cluster of the target feature is the distance between their clusterheads is less than θ. As indicated in (63), an optimal value for this threshold is 3, as he obtained in an experimental evaluation.

In this way, the algorithm merges the cluster associated with *phone* and the one whose clusterhead is *buy*, as their clusterhead distance is 2. Finally, the obtained set os words are:

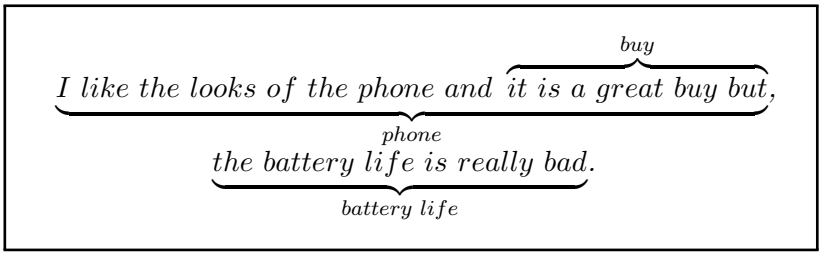


Figure 5.6: Context of the target aspect *phone*, and the aspects *buy* and *battery life*.

To sum up, the context detection algorithm makes use of the graph of dependencies in order to calculate the set of words that express an opinion regarding an aspect. As the words that are detected by the algorithm are very syntactically close to the aspect, there are less noise introduced in the posterior sentiment analyzer, as word that are not related to the aspect are discarded.

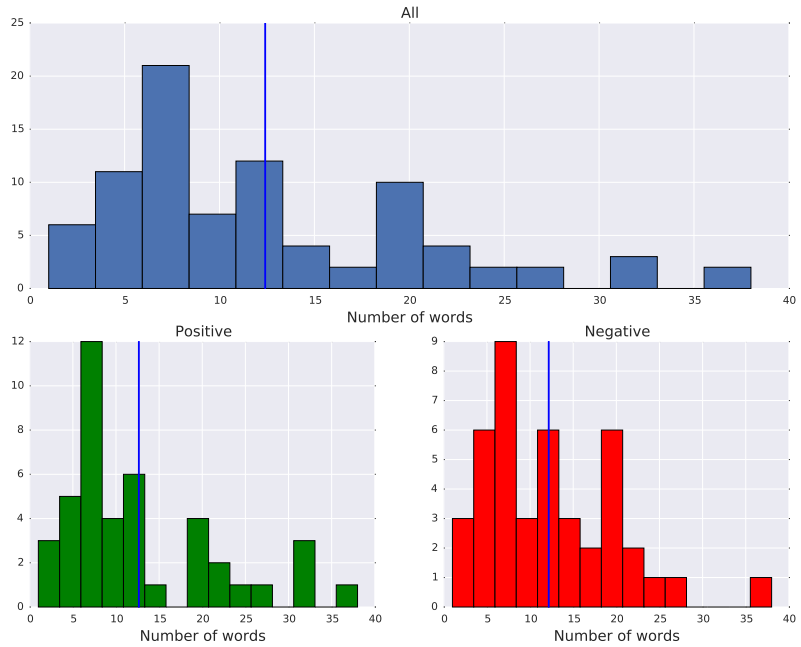
Training

In relation to the training of the sentiment analysis models, we have used a different approach as the one in Chapter 4. In this use case, we use the collected dataset from the Amazon review data, described in 5.1. The difference relies in the length of the dataset. Normally, the tweets consists of one or two short sentences. Nevertheless, in this review dataset, the documents are normally longer. As the convolutional functions used to created the document vector could not work well with longer documents, we split the reviews into sentences. In this way, we train the models with this sentences. As for the polarity, we use the original Amazon rate from the dataset.

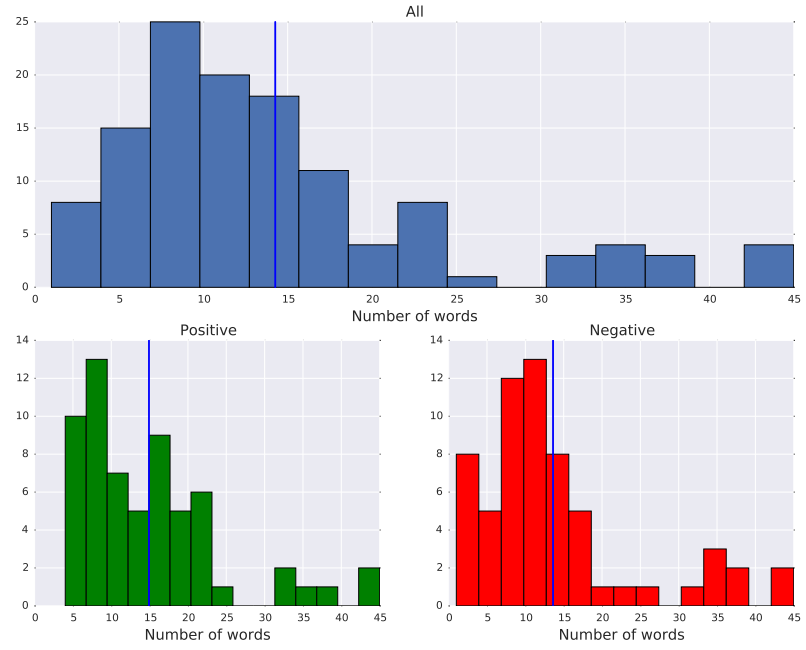
For the training of the *word2vec* model, we use the unlabeled data from this Amazon collected dataset.

The distribution of the number of words of the detected contexts (Sec. 5.2) is illustrated in Figure 5.7. This Figure shows the histogram for the two mobile phone datasets, for each sentiment and also for the aggregated. Also, the average value is represented a by a vertical line. We can observe that the context of the reviews are generally short, and that the mean value for the number of words is 13. That is, each context is composed, in average, by 13 words. In this way, we can observe that the length of the contexts detected from the review

datasets and the length of the Twitter comments are similar. This fact helps in order to adapt the proposed models into this new context.



(a)



(b)

Figure 5.7: Number of words distribution by sentiment polarity and aggregated for mobile phone 1 (a) and mobile phone 2 (b).

Evaluation

Regarding the evaluation of the sentiment analysis system, we have utilized as test datasets the ones described in Section 5.1. These datasets, consequently, have not been used for training purposes. The metric used for the evaluation is the same as the used in Chapter 4, the macro averaged F1-Score. And, in the same way we described in that Chapter, the improvement against the baseline is measured, verifying whether the proposed models surpass this performance or not. Table 5.2 shows the F scores for each classifier in the test datasets.

The results are discussed, firstly, in relation to the base classifiers in Sec. 5.4.1, and then with regards to the proposed models (Sec. 5.4.2). Also, in Sec. 5.4.3 a statistical analysis is applied to the results.

	Mobile phone 1	**Mobile phone 2**
sentiment 140	45.33	46.51
CoreNLP	65.93	61.26
WSD	67.39	62.32
vivekn	62.07	64.57
pattern	67.96	72.85
TextBlob	67.98	75.67
M_G	70.27	69.18
CEM_{SG}^{Vo}	72.16	**73.47**
CEM_{SG}^{MeL}	51.60	63.57
M_{SG}	**73.27**	69.74
M_{GA}	59.09	60.50

Table 5.2: Macro averaged F-Score of all the sentiment classifiers in the phone reviews test datasets. Bold metrics are the best in each test dataset.

Performance of individual classifiers

In general, the performance of the base classifiers has decreased. While in the previous experiment the mean performances by dataset were 73.02, 63.79, 75.32 and 71.81 %; in this case study the mean performances are 62.78 and 63.86 % in the Mobile phone 1 and Mobile phone 2 datasets respectively. That is, a mean F score of 70.98 (Chapter 4) against 63.32 %.

This decrease can be due to the bad adaptation of the majority of the base classifiers to the new domain. A clear example is the sentiment140 classifier: it has been trained specifically for the Twitter domain (50), and consequently it performs badly on the phone review domain. Nevertheless, the CoreNLP improves its performance compared to that of the previous experiment. In the phone reviews domain, this classifier has a mean F score of 63.59 %, while in the Twitter domain its mean performance is 52.44 %. That is, a mean improvement of 11.15 %. The better F score performance among the base classifiers is achieved by TextBlob in both test datasets.

Performance of ensemble classifiers

In this case study the proposed ensemble of both classifiers and features effectively surpass the performance of the deep learning baseline. Nevertheless, in this experiment there are two differences: the affect word vectors and the meta learning ensemble.

As it has been described in Chapter 4, the selected affect word embeddings are trained specifically for the sentiment analysis task on the Twitter domain. In this study, we use them for the reviews domain, resulting in poor results from the ensemble methods that involve this type of feature. In this way, we have omitted the results from the classifiers that rely on the ensemble of classifiers and features which use affect word vectors, as we consider them not relevant. We have, although, included the most basic affect vector model performance results, in order to illustrate the poor adaptation capacity of these vectors.

With respect to the meta learning approach, Table 5.2 shows that it performs poorly than the voting scheme based ensemble. This situation can be explained attending to the size of the meta learner train dataset, as it is lower than the used in the Twitter domain experiment. In the previous experiment, we used a subset of the Sentiment140 dataset to train this meta classifier, while in this case study a subset of the collected train dataset is used.

Statistical analysis

The Friedman ranks can be computed on the results from the previous section. This rank can be used to compare the relative performance of these approaches. They are summarized in Table 5.3.

$\mathbf{M_G}$	$\mathbf{CEM_{SG}^{Vo}}$	$\mathbf{CEM_{SG}^{MeL}}$	$\mathbf{M_{SG}}$	$\mathbf{M_{GA}}$
3	**1.5**	4.5	**1.5**	4.5

Table 5.3: Friedman average ranks R_j for the models relevant in this case study.

In this study, the α value is set to 0.05. Attending to the ranks, the Friedman statistic

$$\chi_F^2 = \frac{12 \cdot 2}{5 \cdot 6}\left((3^2 + 1.5^2 + 4.5^2 + 1.5^2 + 4.5^2) - \frac{5 \cdot 6^2}{4}\right) = 7.2$$

$$F_F = \frac{1 \cdot 7.2}{2 \cdot 4 - 7.2} = 9.0$$

and the critical value $F(k-1,(k-1)(N-1)) = 6.39$. As $F_F > F(4,4)$, we can reject the null-hypothesis. That is, not all the classifiers are the same.

Unfortunately, with the current data it is not possible to apply the Bonferroni-Dunn post-hoc test, due to that the critical difference is too high ($CD = 2.241 \cdot \sqrt{\frac{5 \cdot 6}{6 \cdot 2}} = 3.54$). Consequently, we cannot statistically assure than any of these models perform significantly better than the baseline.

Nevertheless, we can attend to the Friedman ranks and observe that these values align with the previous conclusions. Firstly, affect word vectors cannot be directly used in new domains. Secondly, that the meta learning model needs relatively high data volume to perform well. Finally, the voting scheme and the combination of surface and generic features surpass the performance of the baseline. This can be because of the surface based models adapt better to different contexts, as the characteristics they extract do not differ much in cross domains applications. For example, the word *bad* is, with high probability, a negative word in any domain.

Still, there are word that cannot be directly adapted from one domain to the other. As an illustration of this situation, in two similar domains, drink review and food review, the *cold* word can have different sentiment polarities. In the context of *cold coke*, it has a

positive sentiment, while in the context of *cold pizza* it has a negative connotation. These characteristics are difficult to transfer from one domain to other. Usually, as indicated in Sec. 2.1, this type of information is manually crafted in a work intensive process.

CHAPTER 6

Conclusions and future work

This final chapter presents the conclusions of this master thesis, as well as the future work that can be initialized from this work. The conclusions offer a brief summary of the work developed during this master thesis, pointing out the main results that were obtained. As for the future work, some lines of research are outlined, with special attention to the neural network field, in which many of this master thesis' work is inspired.

This chapter presents the conclusions of this master thesis. After the conclusions, which include the answers to the questions arisen in Chapter 1, the future lines of work that this master thesis leaves open are briefly described.

Conclusions

This master thesis proposed several models where classic hand-crafted features are combined with automatically extracted embedding features, as well as the ensemble of analyzers that learn from these varied features. The different types of combinations aim to effectively aggregate the information captured by the different features and/or classifiers, obtaining a more robust model than the isolated component approaches. The proposed models can be very different, with many elements in its architecture, and some sort of classification can result interesting.

In this way, in order to classify these different approaches, we also proposed a taxonomy that defines two dimensions with which to structure the ensembles of classifiers and features. The classification attends to two main characteristics that the proposed models and many in the literature have in common. That is, the *ensemble strategy* and the *type of features* involved. With these two axis, the taxonomy can be represented in a table, as in Table 2.2. We have classified both the proposed models and the ones found in the literature.

Furthermore, with the aim of evaluating the proposed models, a deep learning baseline was defined, and classification performances were compared with the one of the baseline. With the intention of conducting a comparative experimental study, four public datasets were used for the evaluation of all the models, as well as six public sentiment classifiers. Finally, we conducted a statistical analysis in order to empirically verify that combining information from varied features and/or analyzers is an adequate way to surpass the sentiment classification performance. Besides, we also conducted several additional experiments that are aimed to verify the utility of the word embeddings, and the performance of model that computes the word vectors. Trough this evaluation, we empirically verified that the proposed models significantly surpass the baseline performance.

Added to this, this master thesis also presented a case study where the proposed models are tested in a different domain: reviews oriented to the opinion about a certain product. Also, in this line of work, a new level of sentiment analysis was briefly explored: Aspect Based Sentiment Analysis. Given these new characteristics of the analysis, it has been necessary to add a new technique. That is, context detection of given aspects. This new technique was tackled by the use of an algorithm based on the dependency tree of the

sentence and the computation of distances in the resulting graph. In this way, in this case study we combined the context detection algorithm with the proposed sentiment models in a sentiment analysis case with more granularity, and in a different domain. We empirically verified that the proposed models, combined with the context detection algorithm, perform fairly well in this new domain and sentiment analysis level.

Regarding the first question at the beginning of the paper, we asked whether there is an existing approach to ensemble traditional and deep techniques, and how such an approach would work. To the best of our knowledge, our proposal of a taxonomy and the resulting implementations are the first work to tackle this problem for sentiment analysis.

The second question is whether the sentiment analysis performance of a deep classifier can be leveraged when using the proposed ensemble of classifiers and features models. Observing the scores table and the Friedman ranks (Table 4.3), we see that the proposed models generally improve the performance of the baseline. This indicates that the combination of information from diverse sources such as surface features, generic and affect word vectors effectively improves the classifier's results in sentiment analysis tasks.

Lastly, we raised the concern of which of the proposed models stand out in the improvement of the deep sentiment analysis performance. In this regard, the statistical results point out the $\text{CEM}_{\text{SGA}}^{\text{MeL}}$ and M_{GA} models as the best performing alternatives. As expected, this models effectively combine different sources of sentiment information, resulting in a significant improvement with respect to the baseline. We remark the M_{GA} model, as it does not involve an ensemble of many classifiers, but only a classifier that is trained with an ensemble of deep features.

Future Work

An important aspect of the proposed models is the efficient use of the features, and their dependency with these features. For example, the models that use affect word vectors heavily depend on the capacity of those vectors to retain sentiment information originally present on the represented text. Of course, this fact can be extrapolated to all kinds of features. In consequence, feature quality should be improved in order to increase model performance. In this line of thinking, we believe that many lines of future work lie in the generation of affect word vectors through the use of deep neural networks. The set of techniques related to neural networks allows to automatically obtain the features that will be used as features.

Besides, we are also interested in integrating the proposed models into a unified model that can be expressed in terms of a neural network. In this way, both the feature generator and the classifier model will be united into a single, more robust model.

Moreover, we also are interested in applying the proposed models to the task of aspect based sentiment analysis, in other domains that the one studied in this master thesis. In this way, we intend to generalize the proposed techniques into more complete sentiment analysis frameworks that will perform well in a big variety of domains and sentiment analysis levels.

Furthermore, we intend to extend the domain of the proposed models to other languages, the majority of the work found in this are is in the English language. In special, we would like to expand the work of this master thesis to Spanish.

Finally, an additional future work is the extension of the proposed models to a new task: Emotion analysis. Emotion analysis is an important subtask of opinion mining where the sentiment is not the objective, but the emotions are. That is, the aim of emotion analysis is to discover the emotions enclosed in a text. Some emotions that are usually considered are: *joy*, *feat*, *anger* and *sadness*. The adaptation to emotion analysis would require major changes in the way the models are trained, but not in the models architecture itself.

APPENDIX A

Scientific Python environment

Throughout all this master thesis, the programming language used for the implementation of the processes, models and algorithms is Python [1]. This Appendix gives an overview of the datascience Python environment, briefly describing all main components of this software ecosystem.

The main components used in this master thesis are (all these libraries will be described next):

- Numpy
- Pandas
- Scipy
- Matplotlib
- IPython
- Scikit-learn
- Gensim

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

Numpy

Numpy[2] is the fundamental package for scientific computing with Python. It is used by the majority of the rest of libraries. Numpy offers:

- Support for n-dimensional array objects.
- Broadcasting functions that transparently adapt dimensions between arrays.
- Integration of C/C++ and Fortran code.
- Utilities enclosed in linear algebra, Fourier transform and random number capabilities.

As many machine learning algorithms, the ones used in this master thesis can be expressed as operations between *vectors* and *matrixes*. For this reason, numpy has been of the utmost utility. The numpy capabilities of managing vectors, matrices and higher-dimensional arrays, as well many commonly used operations are basic for this work.

Many frequently used operations in this master thesis can be expressed in numpy with one or a few lines. For example, in order to compute aggregation of features through the convolutional functions (Sec. 3.1), the code can be written as follows.

Listing A.1: "Computing convolutional functions"

```
import numpy as np

# w is the array the contains
# the word vectors of a certain document

np.concatenate((np.max(w,axis=0),
        np.min(w,axis=0),
        np.average(w,axis=0)))
```

[2] http://www.numpy.org/

Pandas

Pandas[3] is a library that provides high-performance, easy-to-use data structures and data analysis tools. Pandas is aimed to be used in data analysis and modeling. In this master thesis, pandas is used to structure the datasets into unified arrangements, as well as performing high performance operations over whole datasets.

Some of the highlights of the pandas libraries are:

- A DataFrame object that allows fast and efficient data manipulation with indexing capabilities.
- Tools for writing and reading data in a huge variety of formats (CSV, Excel, SQL, HDF)
- Intelligent data alignment and reshaping of data.
- Index and label-based sciling, indexing and subsetting of large datasets.
- High performance merging and joining of data sets.

Pandas has been useful in this master thesis when applying operations on whole data sets, or when working with multiple data sets in the same environment. As a brief example, a textual data set can be easily processed in pandas with the following code. This code realizes a basic cleaning and tokenization of text.

Listing A.2: "Basic text munging"

```
import pandas as pd

# df is a dataframe that has preprocessed text
# in the column name text

df['text'].str.strip().apply(lambda x: x.split())
```

[3] http://pandas.pydata.org/

Scipy

SciPy[4] is a Python-based ecosystem of open-source software for mathematics, science, and engineering. Also, scipy is a library that offers fundamental software for scientific computing. In this section we refer to that last meaning of scipy.

Scipy is a collection of mathematical algorithms and convenience functions built on numpy. A python session with scipy becomes a data processing and system prototyping environment rivaling such as MATLAB, Octave and R.

In this master thesis, scipy has not been as intensely used as numpy and pandas, but it has been very important for some key functionalities.

Matplotlib

Matplotlib[5] (66) is the 2D plotting basic library which produces publication quality figures. This library offers a complete environment for plotting. Matplotlib can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc.

On top of matplotlib, there is an additional library called seaborn[6]. Seaborn provides a high-level interface for drawing attractive statistical graphs. Also, it supports both numpy and pandas, as well statistical routines from scipy.

In this master thesis all the plots shown have been generated using matplotlib and seaborn. The integration of both matplotlib and numpy has been specially useful when plotting word vectors and vectors that represent the evolution of a certain performance metric with a hyper-parameter.

IPython

IPython[7] provides a rich framework for interactive computing, allowing the user to execute short black of code and see the results in the same window. IPython is highly integrated with the rest of the scientific python stack, and it is usually use in conjunction. This library also offers an interactive shell, the use of GUI toolkits, and easy to use parallel computing.

[4]https://www.scipy.org
[5]http://matplotlib.org/
[6]https://web.stanford.edu/ mwaskom/software/seaborn/
[7]http://ipython.org/

Also, IPython is part of a bigger project called Jupyter[8]. Jupyter offers the Notebook: a web application that allows to create and share documents that contain live code, equations, visualizations and explanatory text. Between the uses of the Jupyter notebooks, we can find data cleaning and transformation, numerical simulation, statistical modeling and *machine learning*. Finally, Jupyter allows the use of other languages, not only python. Some examples are Ruby and Julia.

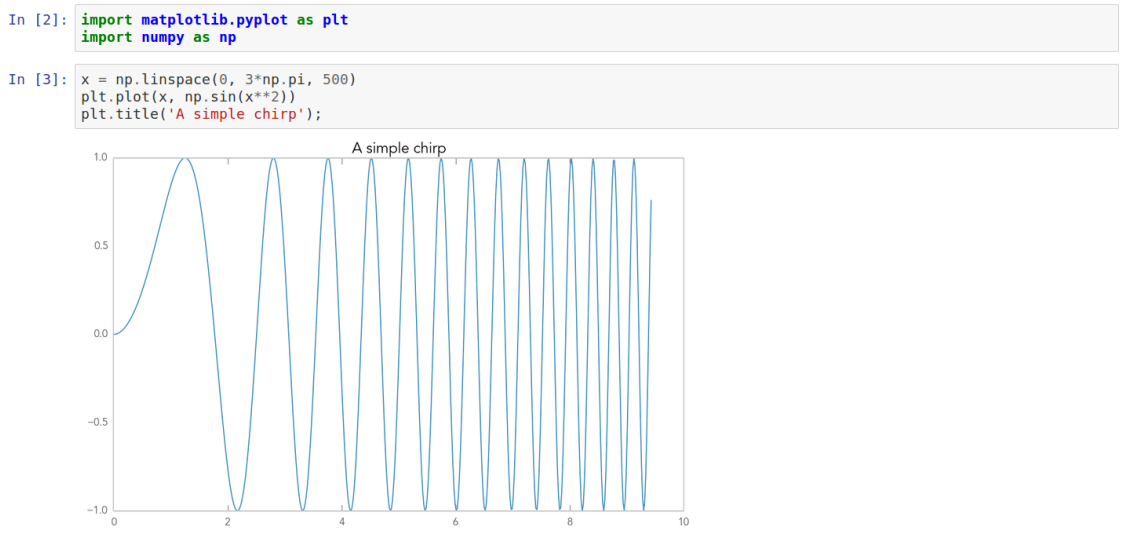


Figure A.1: Example of the IPython interface, and its clean integration with matplotlib.

Scikit-learn

Scikit-learn[9] is a machine learning library for python. It has all the common algorithms and techniques used in machine learning analysis, all under a unified and consistent interface. Scikit-learn implements both regression and classifications algorithms, pre-processing techniques, cross validation methods, etc. Scikit-learn is:

- Simple and efficient tools for data mining and data analysis.
- Accessible to the public, and reusable in different contexts.
- Build on numpy, scipy and matplotlib.
- Open source.

[8] https://jupyter.org/
[9] http://scikit-learn.org/

This library has been used intensively during all the development of the project. With scikit-learn, we have implemented all the proposed models. In particular, we have used very frequently the implementation of the logistic regression algorithm, the cross validation techniques, the performance metrics computation and the reduct of dimensionality algorithms.

The unified interface makes this library very easy to use. For example, all algorithms that need to be trained implement the *fit* method, which takes as parameters the data (and the labels if necessary) to train the algorithm. This methods is implemented independently if is the case of a classifier or a dimensionality reduction algorithm.

Gensim

Gensim[10] (51) is a topic modelling library for humans. Gensim claims to offer scalable statistical semantics, to be able to analyze plain-text documents for semantic structure, and to retrieve semantically similar documents. Also, gensim is scalable and has an efficient implementation.

In this project, gensim has been used because of its implementation of the word2vec algorithm. Thanks to the several optimizations made to this implementation, gensim's word2vec performance is close to the original C implementation[11]. This implementations makes and efficient use of multi-core machines, which is a huge feature. In the following listing, an example of how a word2vec model can be trained on a collection of words, and how an example of its similarity computing capacities are shown.

Listing A.3: "Basic text munging"

```
# train the word2vec model
model = Word2Vec(sentences, size=100, window=5, min_count=5,
   workers=4)

# example of king - man + woman

model.most_similar(positive=['woman', 'king'],
  negative=['man'])
```

[10] https://radimrehurek.com/gensim/index.html
[11] http://rare-technologies.com/parallelizing-word2vec-in-python/

```
[('queen', 0.50882536), ...]
```

APPENDIX B

Linear Regression

In classification, the goal is to use an object's characteristics (or features) to identify which class it belongs to. A linear classifier achieves this by obtaining a linear decision boundary, based on the value of a linear combination of the features input features. These features values can be represented as feature vectors

$$x^{(i)} \in \mathbb{R}^n$$

being n the number of features. The data set is given as a set of feature vectors

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(m)}, y^{(m)})\}$$

where m is the number of examples in the data set. Each label is a real value $y^{(i)} \in \{0, 1\}$, which represents the two different classes (in a binary classification problem).

In this type of classification, the input feature vector x is fed to the classifier, obtaining an output $h_\theta(\mathbf{x}; \boldsymbol{\theta}, b)$. This h_θ function is the hypothesis, and it estimates the probability for each class:

$$h_\theta(\mathbf{x}; \boldsymbol{\theta}, b) = P(y = 1|\mathbf{x}; \boldsymbol{\theta})$$

taking into account that

$$P(y = 1|\mathbf{x}; \boldsymbol{\theta}) + P(y = 0|\mathbf{x}; \boldsymbol{\theta}) = 1$$

Normally, the hypothesis function is expressed as

$$h_\theta(\mathbf{x}; \boldsymbol{\theta}, b) = \sigma(\boldsymbol{\theta}^T \mathbf{x} + b)$$

where the g function is the sigmoid function

$$\sigma(z) = \frac{1}{1 + \exp^{-z}}$$

and $\boldsymbol{\theta}$ is a real vector of weights. The weight vector is learned from the set of labeled training samples. Finally, the class can be decided by applying the following rule:

$$y = \begin{cases} 1 & \text{if } h_\theta(\mathbf{x}) \geq 0.5 \\ 0 & \text{if } h_\theta(\mathbf{x}) < 0.5 \end{cases}$$

The θ parameters are learned by gradient descent, and a common cost function used in this type of classifiers is

$$\sum_{i=1}^{m} L(y^{(i)}, h_\theta(\mathbf{x}; \boldsymbol{\theta}, b)) + \lambda R(\boldsymbol{\theta})$$

where L is the loss function and R is the regularization term. The regularization term is used (67) for avoiding overfitting. In this way R controls the complexity of the fitted function. The expression used can be different, but a common realization of the loss and regularization terms are

$$L(y^{(i)}, h_\theta(\mathbf{x}; \boldsymbol{\theta}, b)) = -\frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(\mathbf{x}) (1 - y^{(i)}) \log(1 - h_\theta(\mathbf{x}) \right]$$

$$R(\boldsymbol{\theta}) = \frac{1}{2m} \boldsymbol{\theta}^T \boldsymbol{\theta}$$

The process of fitting a linear function of the logistics regression can be also be seen as computing the decision boundary as the set of $\mathbf{x}$ for which $P(y = 1|\mathbf{x}) = P(y = 1|\mathbf{x}) = 0.5$. This is given by the hyper plane

$$\mathbf{x}^T \boldsymbol{\theta} + b = 0$$

On the side of the hyper plane for which $\mathbf{x}^T \boldsymbol{\theta} + b > 0$, inputs are classified as 1's, and on the opposite side they are classified as 0's. The bias parameter b shifts the decision

boundary on a fixed amount. If $b = 0$, the decision boundary goes through the coordinate origin. In n dimensions, the space of vectors that are perpendicular to $\boldsymbol{\theta}$ occupy a $n - 1$ dimensional hyper plane.

As an example of how the logistic regression works, Figure B.1 shows en example of a dataset linearly separable, and the decision function of a logistic regression algorithm trained on this dataset. As can be seen, the algorithm successes on obtaining a decision function that separates the two classes. The decision function obtained is:

$$\boldsymbol{\theta}^T \mathbf{x} + b = 0$$

$$-1.713\, x_1 + 1.655\, x_2 - 0.105 = 0$$

With the $\boldsymbol{\theta}$ vector being the orthogonal vector to the decision function.

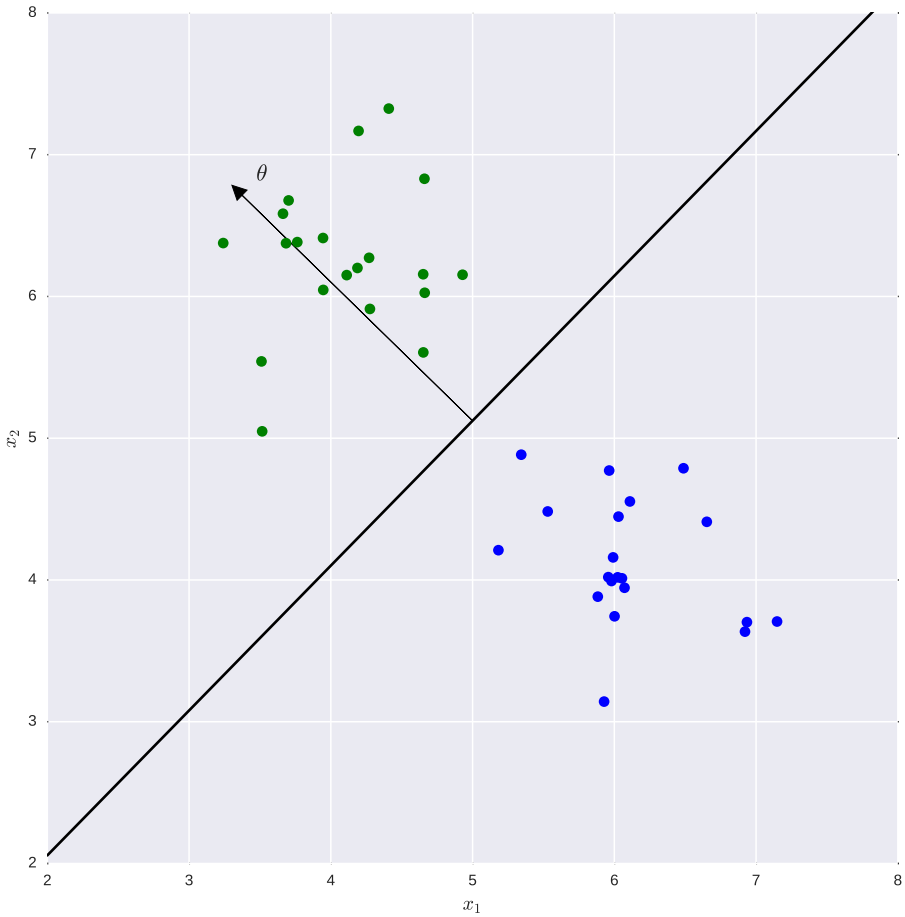


Figure B.1: Decision boundary computed by a logistic regression algorithm trained on the data points.

APPENDIX C

Principal Component Analysis (PCA)

In many machine learning problems data is often high dimensional, and it can be useful to reduce said dimensionality. This reduction can be necessary in order to improve the time performance of a classification algorithm. Or, as is our case, a dimensionality reduction is mandatory for drawing the data in a low dimensional space humans can observe.

If data lies close to a linear subspace, we can accurately approximante each data poiny by using vectors that span the said linear subspace (67). In this cases, the goal is to discover a low-dimensional coordinate system in which the original data can be approximately represented. Generally, the approximation for datapoint $\mathbf{x}$ is expressed as

$$\mathbf{x}^{(i)} \approx \sum_{j=1}^{k} z_j^{(i)} \mathbf{b}^{(j)} \equiv \mathbf{z}^{(i)}$$

Here the i-th vector is approximately represented by the linear combination of the basis vectors $\mathbf{b}^j$ that span the linear subspace (also known as "principal components"). The combination parameters $z_j^{(i)}$ are the lower dimension vector components of the i-th low-dimensional approximation. For a dataset of dimension n, the goal is to describe the data using only a lower number $m << n$ of coordinates $\mathbf{z}$.

The routine for PCA is based on the Singular Value Decomposition (SVD) method, and

it is described in Algorithm 2. In this description of the algorithm, $\mathbf{X} \in \mathbb{R}^{m \times n}$ is the data matrix.

Algorithm 2 Dependency extraction algorithm

(i) Compute co-variance matrix:

$$\mathbf{C} = \frac{1}{m} \mathbf{X}^T \mathbf{X}$$

(ii) Compute the SVD of $\mathbf{C}$

$$\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where $\mathbf{U}^T\mathbf{U} = \mathbf{I}_m$, the columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{C}$ and $\mathbf{\Sigma}$ is a diagonal matrix whose values are the eigen values of $\mathbf{C}$.

(iii) The k first columns of $\mathbf{U}$ are selected, composing the matrix $\mathbf{U}_{\text{reduced}} \in \mathbb{R}^{n \times k}$.

(iv) The lower-dimensional representation of the i-th data point $\mathbf{x}^{(i)}$ is given as:

$$\mathbf{z}^{(i)} = \mathbf{U}_{\text{reduced}}^T \mathbf{x}^{(i)}$$

(v) The approximate reconstruction of the original datapoint is:

$$\mathbf{x}_{\text{approx}}^{(i)} \approx \mathbf{U}_{\text{reduced}} \mathbf{z}^{(i)}$$

(vi) The averaged squared projection error over all the training data made is:

$$\frac{1}{m} \sum_{i=1}^{m} ||\mathbf{x}^{(i)} - \mathbf{x}_{\text{approx}}^{(i)}||^2 = (n-1) \sum_{j=k+1}^{m} \lambda_j$$

where $\lambda_{k+1} \ldots \lambda_m$ are the eigenvalues discarded in the projection.

Typically, in order to choose a k value, the total variation in the data is considered:

$$\frac{1}{m} \sum_{i=1}^{m} ||\mathbf{x}^{(i)}||^2$$

The rule taken is to select the smallest k value so that

$$\frac{\mathbf{x}_{\text{approx}}^{(i)}||^2}{\frac{1}{m}\sum_{i=1}^{m} ||\mathbf{x}^{(i)}||^2} \leq 0.01$$

allowing to retain a 99% of the variance of the original data.

In Figure C.1 a realization of the PCA algorithm is showed. The original two-dimensional data (blue) is transformed into a one-dimensional data (green), using the eigenvector represented as a big arrow. The orthogonal and smaller arrow is the discarded eigenvector. In this example, the total variance retained with the approximation is 94.9%.

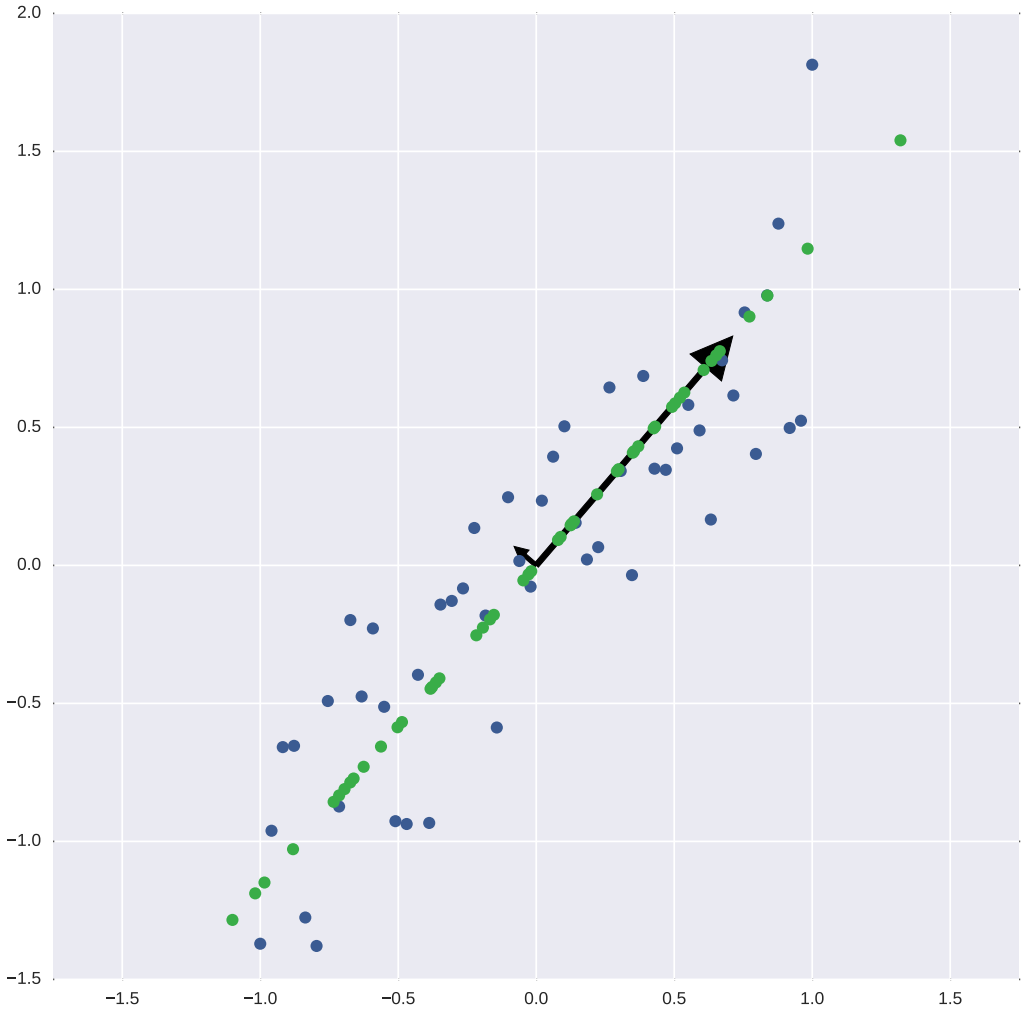


Figure C.1: Example of how the PCA algorithm reduces the dimensionality of the blue data points, resulting in the green data points.

Bibliography

[1] B. Liu, *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press, 2015.

[2] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[3] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86, Association for Computational Linguistics, 2002.

[4] J. Read, "Using emoticons to reduce dependency in machine learning techniques for sentiment classification," in *Proceedings of the ACL student research workshop*, pp. 43–48, Association for Computational Linguistics, 2005.

[5] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pp. 90–94, Association for Computational Linguistics, 2012.

[6] T. Nasukawa and J. Yi, "Sentiment analysis: Capturing favorability using natural language processing," in *Proceedings of the 2nd international conference on Knowledge capture*, pp. 70–77, ACM, 2003.

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[8] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[9] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[11] R. Xia and C. Zong, "Exploring the use of word relation features for sentiment classification," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, (Stroudsburg, PA, USA), pp. 1336–1344, Association for Computational Linguistics, 2010.

[12] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining.," in *LREc*, vol. 10, pp. 1320–1326, 2010.

[13] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith, "Part-of-speech tagging for twitter: Annotation, features, and experiments," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, (Stroudsburg, PA, USA), pp. 42–47, Association for Computational Linguistics, 2011.

[14] E. Kouloumpis, T. Wilson, and J. D. Moore, "Twitter sentiment analysis: The good the bad and the omg!," *Icwsm*, vol. 11, pp. 538–541, 2011.

[15] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins, "Globally normalized transition-based neural networks," *CoRR*, vol. abs/1603.06042, 2016.

[16] P. Melville, W. Gryc, and R. D. Lawrence, "Sentiment analysis of blogs by combining lexical knowledge with text classification," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, (New York, NY, USA), pp. 1275–1284, ACM, 2009.

[17] S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," *Journal of Artificial Intelligence Research*, pp. 723–762, 2014.

[18] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Computational linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[19] L. Polanyi and A. Zaenen, *Computing Attitude and Affect in Text: Theory and Applications*, ch. Contextual Valence Shifters, pp. 1–10. Dordrecht: Springer Netherlands, 2006.

[20] G. Qiu, B. Liu, J. Bu, and C. Chen, "Expanding domain sentiment lexicon through double propagation.," in *IJCAI*, vol. 9, pp. 1199–1204, 2009.

[21] S. M. Mohammad, S. Kiritchenko, and X. Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," *CoRR*, vol. abs/1308.6242, 2013.

[22] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment analysis of twitter data," in *Proceedings of the workshop on languages in social media*, pp. 30–38, Association for Computational Linguistics, 2011.

[23] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis," *Computational linguistics*, vol. 35, no. 3, pp. 399–433, 2009.

[24] A. Sharma and S. Dey, "A comparative study of feature selection and machine learning techniques for sentiment analysis," in *Proceedings of the 2012 ACM Research in Applied Computation Symposium*, pp. 1–7, ACM, 2012.

[25] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification.," in *ACL (1)*, pp. 1555–1565, 2014.

[26] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations.," in *HLT-NAACL*, pp. 746–751, 2013.

[27] D. Zhang, H. Xu, Z. Su, and Y. Xu, "Chinese comments sentiment classification based on word2vec and svm perf," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1857–1863, 2015.

[28] D. Tang, F. Wei, B. Qin, T. Liu, and M. Zhou, "Coooolll: A deep learning system for twitter sentiment classification," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pp. 208–212, 2014.

[29] Z. Su, H. Xu, D. Zhang, and Y. Xu, "Chinese sentiment classification using a neural network tool word2vec," in *Multisensor Fusion and Information Integration for Intelligent Systems (MFI), 2014 International Conference on*, pp. 1–6, Sept 2014.

[30] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 959–962, ACM, 2015.

[31] J. Kim, J.-B. Yoo, H. Lim, H. Qiu, Z. Kozareva, and A. Galstyan, "Sentiment prediction using collaborative filtering.," in *ICWSM*, 2013.

[32] C. Li, B. Xu, G. Wu, S. He, G. Tian, and Y. Zhou, "Parallel recursive deep model for sentiment analysis," in *Advances in Knowledge Discovery and Data Mining* (T. Cao, E.-P. Lim, Z.-H. Zhou, T.-B. Ho, D. Cheung, and H. Motoda, eds.), vol. 9078 of *Lecture Notes in Computer Science*, pp. 15–26, Springer International Publishing, 2015.

[33] D.-T. Vo and Y. Zhang, "Target-dependent twitter sentiment classification with rich automatic features," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pp. 1347–1353, 2015.

[34] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[35] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *EMNLP*, vol. 14, pp. 1532–1543, 2014.

[36] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

[37] L. Rokach, "Ensemble methods for classifiers," in *Data Mining and Knowledge Discovery Handbook* (O. Maimon and L. Rokach, eds.), pp. 957–980, Springer US, 2005.

[38] R. Xia, C. Zong, and S. Li, "Ensemble of feature sets and classification algorithms for sentiment classification," *Information Sciences*, vol. 181, no. 6, pp. 1138 – 1152, 2011.

[39] R. Xia and C. Zong, "A pos-based ensemble model for cross-domain sentiment classification.," in *IJCNLP*, pp. 614–622, Citeseer, 2011.

[40] G. Mesnil, T. Mikolov, M. Ranzato, and Y. Bengio, "Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews," *CoRR*, vol. abs/1412.5335, 2014.

[41] A. Aue and M. Gamon, "Customizing sentiment classifiers to new domains: A case study," in *Proceedings of recent advances in natural language processing (RANLP)*, vol. 1, pp. 2–1, 2005.

[42] V. Sehgal and C. Song, "Sops: Stock prediction using web sentiment," in *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pp. 21–26, Oct 2007.

[43] J. Prusa, T. Khoshgoftaar, and D. Dittman, "Using ensemble learners to improve classifier performance on tweet sentiment data," in *Information Reuse and Integration (IRI), 2015 IEEE International Conference on*, pp. 252–257, Aug 2015.

[44] M. Whitehead and L. Yaeger, "Sentiment mining using ensemble classification models," in *Innovations and advances in computer sciences and engineering*, pp. 509–514, Springer, 2010.

[45] H. Shirani-Mehr, "Applications of deep learning to sentiment analysis of movie reviews," 2012.

[46] K. Zhang, Y. Cheng, Y. Xie, D. Honbo, A. Agrawal, D. Palsetia, K. Lee, W.-k. Liao, and A. Choudhary, "Ses: Sentiment elicitation system for social media data," in *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pp. 129–136, IEEE, 2011.

[47] S. Rosenthal, "Semeval 2014 task 9 description." `http://alt.qcri.org/semeval2014/task9/`, 2014. Accessed on May 30, 2016.

[48] C. Hutto, "Vader sentiment github repository." `https://github.com/cjhutto/vaderSentiment`, 2015. Accessed on May 30, 2016.

[49] H. Saif, M. Fernandez, Y. He, and H. Alani, "Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold," 2013.

[50] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, p. 12, 2009.

[51] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. http://is.muni.cz/publication/884893/en.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[53] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60, 2014.

[54] P. Kathuria, "Sentiment wsd github repository." `https://github.com/kevincobain2000/sentiment_classifier/`, 2015. Accessed on May 30, 2016.

[55] V. Narayanan, I. Arora, and A. Bhatia, "Fast and accurate sentiment classification using an enhanced naive bayes model," *CoRR*, vol. abs/1305.6143, 2013.

[56] T. De Smedt and W. Daelemans, "Pattern for python," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2063–2067, 2012.

[57] S. Loria, "Textblob documentation page." `https://textblob.readthedocs.org/en/dev/index.html`, 2016. Accessed on May 30, 2016.

[58] C. Strapparava, A. Valitutti, *et al.*, "Wordnet affect: an affective extension of wordnet.," in *LREC*, vol. 4, pp. 1083–1086, 2004.

[59] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[60] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *arXiv preprint arXiv:1309.4168*, 2013.

[61] A. Alghunaim, *A Vector Space Approach for Aspect-Based Sentiment Analysis*. PhD thesis, Massachusetts Institute of Technology, 2015.

[62] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177, ACM, 2004.

[63] S. Mukherjee and P. Bhattacharyya, "Feature specific sentiment analysis for product reviews.," vol. 7181 of *Lecture Notes in Computer Science*, pp. 475–487, Springer, 2012.

[64] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, ACM, 2015.

[65] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 43–52, ACM, 2015.

[66] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[67] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.