Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Συγκριτική μελέτη μεθόδων εξόρυξης συναισθήματος σε κριτικές ταινιών** **A comparative study of sentiment analysis techniques on movie reviews domain**
Ονοματεπώνυμο Φοιτητή	Νικόλαος Παναγιάρης
Πατρώνυμο	Γρηγόριος
Αριθμός Μητρώου	ΜΠΠΛ/ 12046
Επιβλέπων	Ευθύμιος Αλέπης, Επίκουρος Καθηγητής

Ημερομηνία Παράδοσης **Οκτώβριος 2016**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Γεώργιος Τσιχριντζής	Μαρία Βίρβου	Ευθύμιος Αλέπης
Καθηγητής	Καθηγήτρια	Επίκουρος Καθηγητής

Abstract

Sentiment analysis has emerged as a field that has attracted a significant amount of attention since it has a wide variety of applications that could benefit from its results, such as news analytics, marketing, question answering, knowledge management and so on. This area, however, is still early in its development where urgent improvements are required on many issues, particularly on the performance of sentiment classification. Document-level sentiment classification aims to automate the task of classifying a textual review, which is given on a single topic, as expressing a positive or negative sentiment. In general, supervised methods consist of two stages: (i) extraction/selection of informative features and (ii) classification of reviews by using learning models like Support Vector Machines (SVM) and Naive Bayes (NB). SVM have been extensively and successfully used as a sentiment learning approach while Deep learning neural networks have been applied only recently , and were not included in comparative studies in the sentiment analysis literature. In this thesis, we survey and implement several deep learning and deep-learning-inspired approaches and we present an empirical comparison between convenient machine learning techniques and Deep learning methods regarding document-level sentiment analysis. We discuss requirements, resulting models and contexts in which both approaches achieve better levels of classification accuracy. Our experiments indicated that SVM outperform the sophisticated DL methods on the benchmark dataset of Movies reviews. Our results have also confirmed some potential limitations of both models, which have been rarely discussed in the sentiment classification literature, like the computational cost of SVM at the running time and DL at the training time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Every day, hundreds of user generated posts are released on the web from newspaper articles to products reviews. However, an important question is how do we make sense of all this abundant information? One of the most exciting applications that motivate us for this research is a more effective processing of the increasing amounts of user-generated content on the web. Under the assumption that the sentiment expressed online represents the general public view, all the abundant information can be computationally used in order an idea from public trends to be conceived. And if the information is analyzed over a period of time, the change in public sentiment on a particular topic through time can be captured.

Sentiment Analysis, or Opinion Mining arose from this need. Sentiment Analysis, is nothing but the computational analysis of people's opinions, sentiment, attitudes and emotions towards a target entity and it attributes [68]. Sentiment analysis encapsulates the following tasks: Feature based Sentiment classification, Sentiment classification and opinion. Main research study fields, in the realm of Sentiment Analysis and opinion mining are: sentiment classification, feature based Sentiment classification and opinion summarization. Sentiment classification bestows with classifying entire documents or text or review in conformity with the opinions towards certain objects. Sentiment classification, can be conducted on different levels of granularity: document, sentence, or aspect levels [30]. In literature exists three different methods that tackles the problem of sentiment analysis: machine learning methods lexical based methods and linguistic analysis.

In this thesis, we address the problem of sentiment classification on IMDB reviews. The objective of this work is to draft a procedure that assigns either a positive (1) or a negative (0) sentiment to a given IMDb movie review. A lot of work has already been done on sentiment analysis for movie reviews using either machine learning methods or linguistics approaching in this thesis we consider only machine learning methods.

Learning good representations for text, such as words, sentences and documents, is essential for natural language processing (NLP) tasks like text classiffcation and sentiment analysis, which has attracted considerable attention from both academic and industrial communities [10]. It is common to use bag-of-words or bag-of-ngram to represent text [23] and train the models based on such representation. However, each word or n-gram is a unique feature and their interactions and the whole word order in the text are not preserved in the text representation, which limits the functionality of the learning models based on such representation. In recent years, the word embedding and neural network models have brought new solutions to learn better representations for NLP tasks. The simplest one is Bag of- Word-Vectors (BOW vector) model. But Landauer [62] estimates that 20% of the meaning of a text comes from the word order.

Therefore, these models are still oversimplified because of the loss of order information. Neural language model [7] was then proposed to leverage word embedding representation to infer the next-word distribution, but it still fails to fully utilize the sequence of the context words.

More recent work has focused on creating vector representations for both words and documents, with the idea of retaining as much information as possible. In Word2Vec [?], vector representations are computed for each word, with the result being that words whose meanings are related are generally closer in terms of Euclidean distance than between unrelated words. On document level, the authors proposed, as an extension, to assign to each text a unique paragraph vectors. They

proposed an unsupervised learning framework that learns continuous vector representation for each document where the document vector representation is trained to be useful for learning word vectors.

Recurrent Neural Networks take words order into account, but later words are more influential on the final text representation than the former words. Generally, in sentiment analysis task the important words are scatter in the document. However, in the movie reviews domain it is believed that the last words in the review are strong indicator of the over polarity expressed in the review.

On the other hand, Convolutional Neural Networks (CNN) are naturally capable of solving this problem due to the use of the max-pooling layer which is responsible for equal treatment of the words [35]. Note, that RNN is more suitable for processing sequentially the information and getting fixed length output whereas the CNNs are using sliding windows with different width and filters in order to perform feature mapping and then polling is used to get fixed length output. Additionally, the simplicity and their efficiency made the CNNs more appealing for text classification tasks, where achieved remarkable results [57]. However, CNNs are far from perfect, especially when it comes to determine the windows width and the too many filters parameters.

All in all, in this thesis we compared as many of the state-of-the art machine learning algorithms exist in literature. We also compare the BoW model with the state of the art features extraction methods, and for our surprise, we found that BOW outperforms more sophisticated methods.

1.1 Research Goal

Based on the observation above we arrived at the following primary research questions.

RQ 1. Which machine learning algorithms perform better in Sentiment analysis?

We evaluate the performance of our models on the Stanford Large Movie Review Dataset (IMDB) dataset [70] that allow us to evaluate and compare performance against existing publications on binary classification task.

RQ 2. Which natural language representation is suitable for Sentiment analysis?

Natural Language representation in vector spaces has been successfully used in many NLP tasks. Previous research has employed vector representations to present the syntactic and semantic information in textual contents. In this thesis, we investigate the effectiveness of vector space representations.

1.2 Thesis Outline

Thesis consists of 7 main chapters starting with an introduction. Section 2,3,4 covers the theory part of the work including feature extraction and classification. Section 5 review of prior research in sentiment analysis. Section 6 covers the implementation part, where the chosen methods are explained and demonstrated . In the final chapter we presented ours research findings.

Chapter 2

Feature extraction

This section describes the theoretical principles and methodologies of the implemented sentiment analysis classification. The main objective of the three theory sections, is to point out the most relevant factors of machine learning techniques in respect of sentiment analysis. Loosely speaking, machine learning aims to recognize complex patters in order to produced intelligent decisions based on a given data set. In terms of text classification this involves three tasks: feature extraction, feature selection and classification.

Feature extraction is the procedure of transforming arbitrary data, intro numerical feature representations usable for machine learning approaches. Feature extraction, is reducing the dimensions of the initial set of data by deriving values about features that are considered informative and non-redundant the so-called feature vectors. This process is known as vectorization. Extracted features contain the relevant information from the input data and will be used for further means instead of the complete initial data.

Feature extraction is the key of the success of the classification results. However, some problems need to be addressed while one performing feature extraction. To begin with, when the training set contains a large number of variables, vectorization techniques requires large amount of memory and computation power. An important pre-step is the preprocessing of the data. Before we begin to describe feature extraction methods used in this thesis, we first analyze the pre-processing techniques are usually performed in Sentiment analysis.

2.1 Text preprocessing

Text Preprocessing is the procedure of cleaning and preparing text data for the vectorization process. The necessity of this steps lies in the fact that online texts contain usually noise and uninformative parts such as HTML tags, scripts, and advertisements. Since each word in the text is treated as one-dimension noise and uninformative words are increasing the dimensionality without having any impact on general orientation of the text. Thus the classification process is becoming more difficult. The difficulty is not only manifested in the robustness of the classification results but also leads in the increase of computational complexity of the process. The preprocessing consists of many steps; depends on which dataset in used or which approach the researcher will follow. Some of the most known steps are online text cleaning, white space removal, stemming, stop words removal, negation handling most of the which are used in this thesis but without improving significantly the classification accuracy.

2.1.1 Online text cleaning

As we mentioned above, Web pages contain along with the main texts and other irrelevant information such as HTML tags (for example $< p > or < b >$). To avoid low classification accuracy problem, the text should be cleaned in order to retain only information of interest. Many ways of text cleaning have been proposed such as "HTML Cleanup" or document object model. In this thesis, we used a python library named Beautiful Soup which parses the HTML features from them

and put them in an tree structure based object in order to be separated from each other. In this way only the core interest text is extracted.

2.1.2 White space and stopword removal

It is not unusual text containing two or more white spaces especially when the HTML tags are removed. Whitespace removal helps to tackle this problem by keeping only one. Stopwords are a fundamental aspect of the natural language. They are considered to be without any discriminant value in the text in terms of sentiment analysis. But this is not the case in this thesis, the Stopwords removal leads to lower accuracy results in all methods used. A discrimination of Stopwords is proposed in [71] in which authors considers either as general or domain specific. One can remove stopwords with many different methods. In this thesis, we use a list of standard Stopwords the so-called stop list. We used Rijsbergen stoplist [102] which is the most commonly used stoplist in natural language processing and it is also provided by python's NLTK library. One other approach we tried, were the words with high frequency to be treat as stopwords and to be removed. However, that approach led to significant decrease of classifiers accuracy.

2.1.3 Stemming

In stemming process, the suffix of the word is stripped and its transformed in its basic or root form. For example, If a text contains words like "fishing, "fished", and "fisher" should not be treated differently in the terms of sentiment analysis as they have the same meaning and don't add different polarity. Stemming them will reduce them to the word 'fish' and the word frequency will be set to 3 instead of three words with frequency one. As all text transformations aims to reduce dimensionality stemming does it on a larger scale. For instance, the support vector machine classifier which is used in this thesis , each word is considered to be a vector of its dimension. Hence stemming does not only reduces the dimensionality but also allows a more expressive weight distribution of word weights through their frequencies.

The most known algorithm ,which was introduced in 1979 , and is used widely in literature is the Porter algorithm or the so-called suffix stripping algorithm. Porter used his algorithm in an information retrieval project [122] and proved that accuracy is significantly increased. Although the vocabulary was reduce 40% percent when we applied porter's algorithm the accuracy dropped 10%.

2.1.4 Handling negation

Handling negation is of high importance task not only for sentiment analysis but also and for the natural language processing field. This is due to the fact that one negative word could change the polarity of the whole sentence. Negations can be expressed in many types. In sentences can be found a direct negation where the negation and negated word create a bigram or the negation can be expressed in long distance from the words that negates and thus to be modeled is almost impossible. However, some time the negation words does not inverse the polarity instead increases its density. Thus, negation handling can be seen as of great importance task in text preprocessing but the exhaustive presentation of methods is used to tackle this problem is beyond the scope of this thesis. In the next paragraph, we describe the method we used to handle negation.

In [84] the author devised and algorithm for handling negation through states variables and bootstrapping. The state variable stores the negation state and then the word followed by a "not" or n't is transformed into "not_" + word. The state variable is reset when a puncation mark is encountered or when there is a double negation. The pseudo code derived from [84] is as follows

2.2 Symbolic feature representations

Symbolic models dominated the field of Natural Language Processing (NLP) in the past. N-gram language modelling, part of speech tagging and parsing are some of the success stories of symbolic models but they are inflexible in the sense that they are unable to capture the fact that the language is constantly growing and changing and that the meaning of language depends on things such as

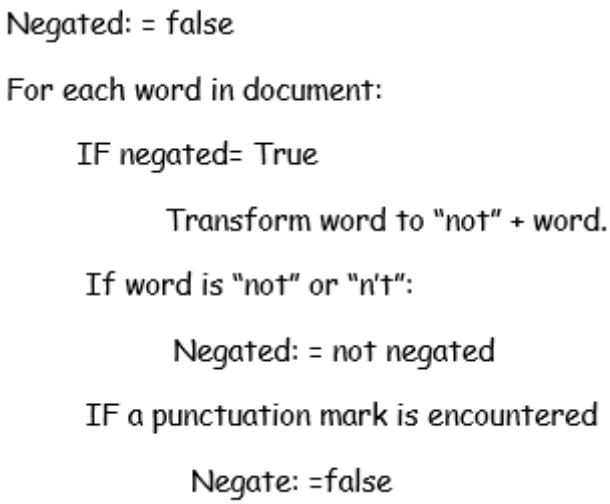


Figure 2.1: Negation Handling pseudocode [84]

the context in which it is used and the topic of conversation. In the following sections we describe Bag- of - Words model and TF-IDF weigthing scheme based on [72].

2.2.1 Bag of words or bag of N-grams

In natural language processing, the most widely used symbolic representation of features, but also quite effective, is the Bag of Words [41]. The main idea behind this method is to simply create a feature for each word that appears in the training texts. That feature for a text vector will then be valued 0 if the word does not appear in the corresponding text.

In the Bag of Word first a string tokenization is performed that assigns an integer identification for each token. For every word in the vocabulary, there is exactly one position in vector, and if a word occurs n times in a text, this component will be n in the vector. The vector including all of the token frequencies for a given document is considered a multivariate sample. For instance, if we tokenize and count word occurrences of text documents ["this is my first sentence", "this is my second sentence", "is this my second sentence"] Each term found by the analyzer during the fit is assigned a unique integer index corresponding to a column in the resulting matrix['first','is','my','one','second','this','sentence'].

The bag-of-words model comes with many variations and extensions the most popular of which is by including word n-grams. Word n-grams are sequenced of n words that can be found in the text. For example, the bigram of "this is a sentence" "this is ", "is a", "a sentence". The reasoning here is that is believed that sequences of words carry their own meaning and therefore classification might be improved. Whether higher-order n-grams are useful features appears to be a matter of some debate. N-grams can also be used for sequences of characters in a similar way but this approach is beyond the scope of this thesis so only word n-grams. Other ways to extend the feature vector can be to include parts of speech, word positions in the text (text at the end of the text could be of more or less importance) and degree modifiers ("too," "enough") [65].

Once the indexing of document is performed and initial word frequencies are calculated various transformation can be performed to summarize or aggregate the extracted information. The Frequencies of words or terms are indicators of how important or not a word in each document is. Therefore, words that occur with great frequency, but not too frequently, considered to be of great importance descriptors of the meaning of the specific document. However, the words count themselves should not be assumed proportional to importance as descriptors of the documents. Thus a transformation should be performed. In this thesis we use the binary bag of word model

that is defined by:

$$f_i(X) = \begin{cases} 1 & if\ d_i\ containt\ word\ w_i \\ 0 & else \end{cases} \tag{2.1}$$

The document-term matrix contains only zero's and one's which indicates presence or absence of words.

Another issue to consider more carefully and reflect in the indices used in further analyses is the relative document frequencies (df) of different words. A common and useful transformation that reflects both the specificity of words (document frequencies) and the overall frequencies of their occurrences (word frequencies) is inverse document frequency (for the i'th word and j'th document):

$$f_i(X) = \begin{cases} (1 + log(wf_ij))log\frac{N}{df_i} & if \quad wf_ij \geq 1 \\ 0\ else & if \quad wf_ij = 0 \end{cases} \tag{2.2}$$

In above formula N is the total number of documents; df_i is the document frequency for the i'th word.

2.2.2 Term frequency & inverse document frequency

Term frequency (TF) indicates the relative importance of a term in a document, which means that the more times a word appears in a document, the more important it is to that document, while TF/IDF weighting scheme denotes how a word is important to a document in a certain type of collection or corpus. For each word in a document, TF/IDF calculates values as the product of the frequency of the term in the document times the inverse document frequency. Words which have higher TF/IDF numbers imply a strong relationship with the document in which they appear. The weight that the model calculates is an indicator of how important a word is to a document in a corpus. The importance is strong correlated with the number of times a word is found to a document but is counterbalanced by its frequency in the corpus. That means that if a words frequency is high in a document but it occurs in less number of documents in the corpus then is clearly a strong indicator for the document meaning. Conversely, if the frequency of a word in a document is high but also can be found in many documents in the corpus that means that word is a common word or a domain Stopword thus the contribution of this word is of minor importance to the classification models. For instance, in our dataset, the word movie or actors is in almost all the documents thus is ignored. Thus to measure the importance of a term t_i within a particular document is given by:

$$tf_{i,j} = \frac{n_{i.j}}{\sum_k n_{k,j}} \tag{2.3}$$

Where $n_{i.j}$ denotes the number of occurrences of the term under examination in document d_j the denominator is the number of all terms in a document d_j

Raw frequencies as described above suffers from a critical problem: considered equally important when it comes to assessing relevancy on a document. But certain terms hold little or no discriminating power in a document. For instance, in movies domain, the word movie is almost in every document. To this end, the inverse document frequency is to scale down the term weights of terms with high collection frequency, defined to be the total number of occurrences of a term in the collection. It is a logarithmically scaled fraction of all documents in corpus that contain the word and is calculated by dividing the total number of documents by the number of documents containing the term and taking the logarithm of that calculation:

$$idf_i = log\frac{N}{dfi} \tag{2.4}$$

In above Equation 2.4, N is the total number of documents in the corpus, denominator is the number of documents where the term t appears. In order to division-by-zero when a term is not in the corpus we adjust the denominator to 1+denominator. As can be seen the idf of a rare term is high, whereas the idf of a frequent term is likely to be low.

Hereby, a combination of term frequency and inverse document frequency is made in order to produce a composite weight for each term in each document the so-called TF/IDF weighting scheme. It is given by:

$$f - idf = tf_{i,d}) \, x \, Idf_t \tag{2.5}$$

The rough idea is that $tf - idf_{t,d}$)assigns to term t a weight in document d that is :

- highest when t occurs many times within a small number of documents (thus lending high discriminating power to those documents)
- lower when the term occurs fewer times in a document, or occurs in many documents;
- lowest when the term occurs in virtually all documents

Thus, a document can be seen as a vector with one component corresponding to each term in the dictionary, together with a weight for each component [72].

Finally, the aforementioned methods although they appeal from their simplicity suffer from major drawbacks. To begin with, the dimensionality of vector space is equal to the vocabulary size which leads to an extremely high dimensional space. Moreover, sparsity is a major problem due to the fact that a text block only contains a small fraction of all words in the vocabulary. Last but not least, all three representations do not capture the semantics of words. For example, the projection of words "fast" and "quick" in vector space is as far as they are from color even though that are semantic similar. Thus the need for convex base approaches, as we mentioned in the introduction of this sections arise. In this thesis we examine the neural distributed representations.

2.3 Neural distributed features representations

Distributional models or VSMs or Semantic Space Models try to overcome some of the inflexibility of the symbolic models by moving away from fixed discrete representations to flexible continuous vector representations of language, where the meaning of a word is derived using its distributional statistics in a text corpus or in more simpler terms, the meaning of a word is derived from its use in the text corpus. In the following paragraphs, a formulation of the the problem of modeling the natural language is given, all the state-of- art methods, to our best knowledge are latter described.

2.3.1 Natural language modeling

In every natural language processing project, the first problem need to be addressed is which model will be used for modeling the language. Below we formally define the language modelling problem.

In language modelling problem a probability distribution p over the sentences, in the language, is to be found in order the following equation to be fulfilled:

$$\begin{gathered} \textstyle\sum_{s \in V^\dagger} p(s) = 1 \\ p(s) \geq 0 \forall s \in V^\dagger \end{gathered} \tag{2.6}$$

Where V is the vocabulary which contains words in the under consideration language, $V^\dagger$ is a collection of all possible token in the language. Thus the probability function can be formulated as:

$$p(s) = p(w_1, ..., w_n) \tag{2.7}$$

Whereas the product rule is given by:

$$p(w_1, ..., w_n) = \prod_{i=1}^{n} p(w_1 | w_1, ..., w_n) \tag{2.8}$$

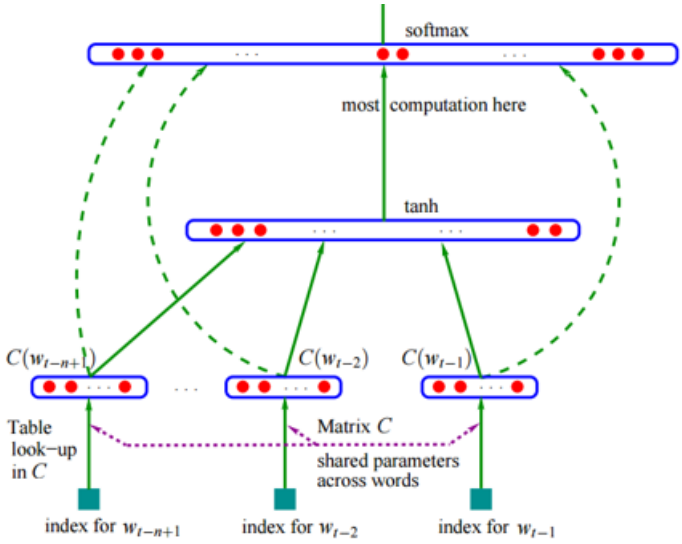


Figure 2.2: Neural probabilistic language mode [7]

Making the assumption that a words depends on previous n words the above equation can be simplified as:

$$p(w_1, ..., w_n) = \prod_{i=1}^{n} p(w_1|w_{i-1}), ..., w_{i-n+1})) \tag{2.9}$$

Thus, in order to build a model that predicts the conditional probability of a word given the previous n words Bengio [7] introduced a neural network as shown in figure 2.2

2.3.2 Neural probabilistic language model (NPLM)

The Neural probabilistic language model that introduced by Bengio has as objective to create a probability function over the distributed word vectors in order to construct a statistical language model. The probability of a word sequence is the product of conditional probabilities of the next word considering the previous ones:

$$P({w^T}_1) = \prod_{t=1}^{T} P(w_t \,|\, w_1^{T-1}) \tag{2.10}$$

The NPLM tries to estimate the probability w_i of the t^th word in the sequence. The neural network responsible for calculating the probability , is a single layer neural network counting only the hidden layer as shown in Figure 2.2

For each input sequence, the neural network outputs a vector $y \in R|V|$where y_i denotes the unnormalized log-probability w_i. The y is given by:

$$y = b + W_x + Utanh(d + Hx) \tag{2.11}$$

Where the hyperbolic tangent is applied to produce non-linearity in the network and

$x = (C(w_{t-1}), \ldots, C(w_{t-n=1}))$ are the concatenated vectors of the index vectors of the previous n words to the word w_t. The output of neural network is given by the Softmax function which converts y log probabilities to positive probabilities summing to 1 as follows:

$$p(w_t \,|\, x) = \frac{e\mathring{y}}{\sum_{i=1}^{\nu} e^{y_i}} \tag{2.12}$$

The learning procedure is based, as we described in section 3, on gradient descent:

$$\theta \leftarrow \theta + e\frac{\partial log(w_t|x)}{\partial \theta} \tag{2.13}$$

Where $\theta = W, U, C$ are the free parameter of the model which we want to optimize in the parameters space by taking small steps of size ε. Thus the optimum word representations are found ($C \in \theta$)

The above network tries to solve the problem of natural language modelling so the word representation is a by-product result. Thus this approach is an indirect method of forming word representations. One major drawback of these techniques is that they are computationally expensive, and thus slow to train and test.

In 2013, Mikolov [78] introduced a recurrent neural network language model (RNNLM) which overcomes the limitation of the above described model. This is due to the fact that, as an RNN architecture it inherits the short term memory of a typical RNN which leads to the modeling of more complex structures than a shallow neural network can also. But it also inherits and the main disadvantage of the RNNs the prohibitively large computational cost. However, the authors succeed to reduce computational complexity by incorporating approximate functions. The result of this are two new models, summarized under the name Word2vec: continuous bag-of-words models (CBOW) and continuous skip-gram models (skip-gram).

However, in 2014 Mikolov [63] extended his previous work by using a paragraph vector. This new type of vector can be seen as a memory to learn word vector from arbitrary block of text by learning and sharing a common vector. The CBOW model that is extended by the paragraph vector (PV) is called PV-DBOW, while the extended Skip-gram model is called Distributed Memory Model of Paragraph Vectors (PV-DM). This two model can be summarized under the name Doc2Vec or Paragraph2Vec.

A modification of skip-gram model was proposed in [35] where the authors used a negative sampling technique along with skip-gram model. In other words, they suggest that Mikolov method factorizing a word-content matrix implicitly whereas they implement this factorization directly. In [93], the authors proposed the so called Global Vector model (Glove.) Their model incorporates a combination of matrix factorization methods and local context window methods. In the next section we describe both Word2Vec and Doc2Vec models. Glove model was not used in this thesis.

2.4 Word2Vec

Like the neural language model, the word2vec models learn vectors known as embeddings by training a network to predict neighboring words. However, the prediction task is not the main task as in NPLM. Words that are semantically similar often occur near each other in text, and so embeddings that are good at predicting neighboring words are also good at representing similarity. The advantage of the word2vec methods is that they are fast.

The massively reduce of computational complexity the Mikolov's models achieved, without sacrificing the quality of the word vectors, is due to the fact that are simple log-linear models as opposed to non-linear NPLM. In fact, the neural network architecture used in this models does not contain a non-linear layer and does not use matrix weighting of the input vectors which are the costliest operation in NPLM. The description of the following models is based on [54].

2.4.1 Skip gram model

The skip gram model as shown in figure 2.3 learns two separate word vector for each word w: the word vector ν and the content vector c. These two vectors are represented by two matrices W and C respectively. Each row i of the word matrix W is the $1 \times d$ vector embedding vi for word i in the vocabulary while C is the flipped version of W.

Let's consider the prediction task. We are given a corpus of length T and examine the t-th word w^t whose vocabulary index is j. The skip gram model predicts each neighboring word in a context window of 2L words from the target word. So for the word w^{t+1}) whose vocabulary index is $k(1 < k < |V|)$. Hence the task is again to calculate the $p(w_k|w_j)$. The core computation that

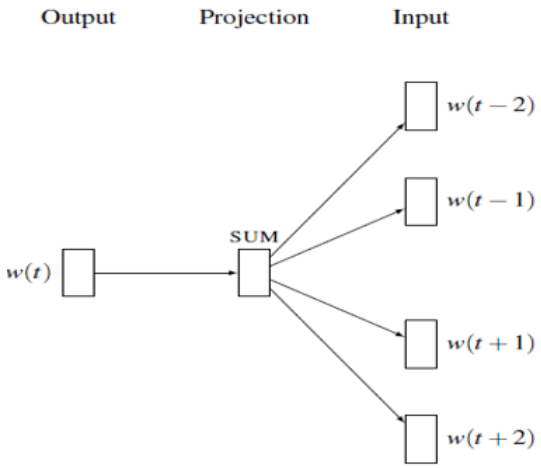


Figure 2.3: Skip gram model for learning vector representations of words [78]

has to be done in order the probability $p(w_k|w_j)$ to be calculated is the dot product between the context vector for w_k and the target vector for w_j. We'll represent this dot product as $c_k \cdot \nu_j$, where c_k is the context vector of word k and ν_j is the target vector for word j. Of course dot product is not probability thus we can use the Softmax function to normalize the dot product into probabilities. Computing this denominator requires computing the dot product between each other word w in the vocabulary with the target word w_i.

$$p(w_k|w_j) = \frac{exp(c_k \cdot \nu_j)}{\sum_{i \in |V|} exp(c_i \cdot \nu_j)} \tag{2.14}$$

In summary, the skip-gram computes the probability $p(w_k|w_j)$ by taking the dot product between the word vector for $j(vj)$ and the context vector for k (c_k), and turning this dot product $c_k \cdot \nu_j$ into a probability by passing it through a Softmax function.

2.4.2 Continuous bag of words (CBOW)

The CBOW model is roughly the mirror image of the skip-gram model. Thus, is a predictive model but the objective this time is to predict the current word w_t from the context window of 2L words for $L = 2$ the context is$[w_{(t-2)}, w_{(t-1)}), w_{(t+1)}, w_{(t+2)}]$ as also shown in the figure 2.4 .

As we mentioned, the objective of CBOW is to predict the word in the middle of c previously encountered words and c following words:

$$maximize_w p(w_t|w_{t-c}, w_{t-(c-1)}, ..., w_{t+1}, w_{t+(c-1)}, w_{t+c}) \tag{2.15}$$

So, the CBOW, like the Skip-gram model, tries to maximize the average log probability as follows:

$$maximize_w \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \geq j \neg 0} Logp(w_t|w_{t+j}) \tag{2.16}$$

2.4.3 Skip gram with negative sampling

In the previous sections, we have already mentioned the objectives of learning word and context vectors matrices.The rough idea described is to make the vector of a word more similar to the vectors of neighboring words rather than the vector of other words.

The prediction algorithm we described before, predicting the probability of a word by normalizing the dot-product between a word and each word in the content by the dot product. Therefore, in order the denominator to be calculated a lot computations need to be performed. Thus, this version of prediction algorithm is computationally expensive.

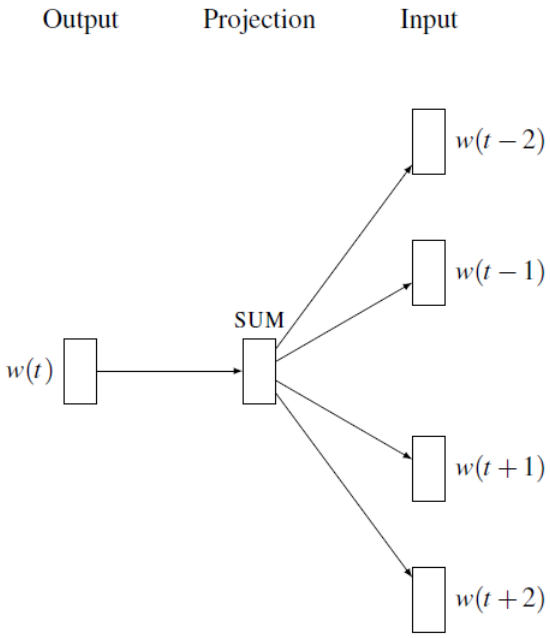


Figure 2.4: Continuous Bag of Words model for learning vector representations of words [78]

In order this problem to be addressed, a new version of the algorithms proposed, that uses negative sampling where they approximate this denominator. In the following few paragraphs we offer a brief sketch of how skip gram model with negative sampling works.

In the original form of the skip gram model, the algorithm scans the corpus, and for each target words choosing the surrounding words. However, when negative sampling is used, the algorithm scans the corpus and for each target word chooses the surrounding context words as positive samples and the non-neighbor words are labelled as negative samples. In other words, this algorithm tries to move the word vector toward the neighbor words without take into account the finite number of noise words. In fact, the objective is to learn a word vector whose dot product which each context word is high and the noise words do product to be low. More formally the objective is as follows:

$$log\sigma(c * w) + \sum_{i=1}^{k} E_{wi\ p(w)}[log\sigma(-w_i * w)] \tag{2.17}$$

Where σ is a sigmoid function of the dot product $\sigma(\chi)1/(1 + e^x)$. The noise words $w_i$are extracted from the vocabulary V according their weighted probability.

The algorithm starts with randomly initialized word vectors and then the documents are processed, adjusting the word vectors so as to maximize the above equation. As in the common neural network, the weights are learned using an algorithm like stochastic gradient descent using error backpropagation in order the gradient to be propagated back through the network.

Nonetheless, although negative sampling is a different objective than the probability objective, and so the resulting dot products will not produce optimal predictions it is shown to produce good vectors.

2.4.4 The skip-gram model as a network

In order for the model to make use of error backpropagation, we should see the procedure of selecting two vectors from W and C matrices as a network in which we can propagate backwards across. In figure 2.5 a simplified visualization of the model is given whereas a single context word is to be predicted in order the use of Softmax function to be shown over the entire vocabulary rather than over the k noise words.

But how the network is computing the same probability as the doc product version? To begin

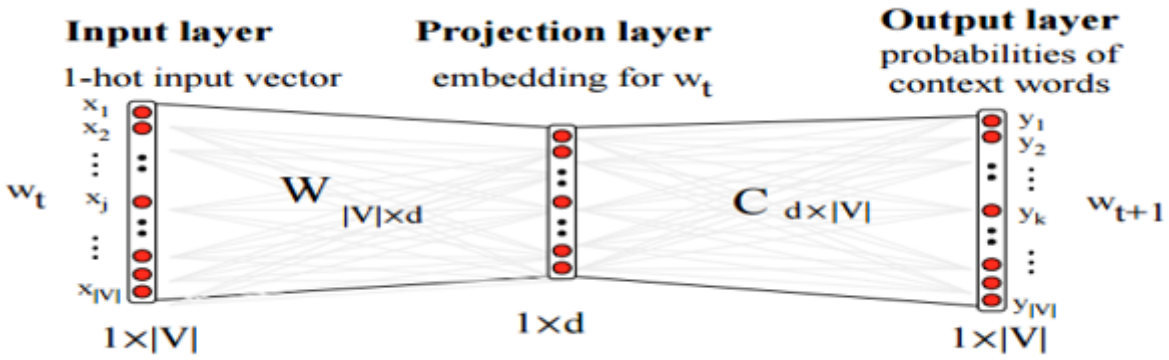


Figure 2.5: The skip-gram model viewed as a network [60]

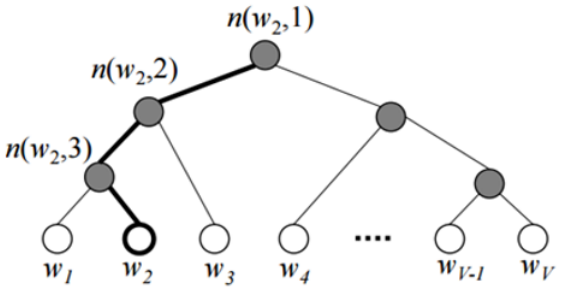


Figure 2.6: An example of binary tree for the hierarchical Softmax mode

with, the input vector x is one hot vector for the current word w_j. As we mentioned above, the one hot vector representation is just a vector where all it elements are zeros except a one which indicates the vocabulary index of the word. In fact, in a one-hot representation for the word $w_j x_j = 1$ *and* $x_i = 0 \forall i \neq j$

The prediction of the probability of each of the 2C output words is a 3 steps procedure:

- In the first step, the input vector x is multiplied by the input matrix W to give the projection layer. Note that, each column of the W is just an embedding for word w_t and the input is a one hot representation for w_j consequently the projection layer for the input x will be $h = v_j$
- In the second step, the dot product between the projection vector h and the context matrix$C$ is computed for each word of the $2C$ context words. The result of this calculation is an output vector that weights each of the vocabulary words $|V|$
- In the last step, the dot product is interpreted into probabilities by using the Softmax function as we saw earlier

2.4.5 Computational optimizations

As we have point out, CBOW and Skip-gram models are computationally inefficient. Thus we described the skip-gram model using negative sampling as one way to optimize its performance. However, in literature are proposed more methods in order the performance to be improved.

In 2005 Morin [82] proposed hierarchical Softmax to speed up the training. This method constructs a Huffman tree (Figure 2.6) to represent all words in the vocabulary. The words are the leaf units of the tree. Thus for each unit of the tree there exists an optimal path from the root to unit that is used to compute the probability of the word represented by that leaf .

The probability of a word to be an output word is defined as:

$$p(w = w_o) = \prod_{j=1}^{L(w)-1} \sigma([\![n(w, j+1) = ch(n(w, j))]\!] * v'^{T}_{n(w,j)} h) \tag{2.18}$$

Where ch(n) is the left child of leaf n and $\nu'_{n(w,j)}$ is the output vector of the inner unit $n_{(w,j)}$

$$Moreover, [\![x]\!] \begin{cases} -1 & if\ x\ is\ false \\ 1 & otherwise. \end{cases} \tag{2.19}$$

2.5 Doc2Vec

Although the semantic word spaces we described in the previous paragraphs are essential in natural language processing their ability to capture the compositionality of human language is limited. This is due to the fact that the word vectors learned cannot be directly used to represent as in our case documents.

Progress towards learning representations of document level that capture semantic compositionality has been made; but most models are using a simple weighted average of words vector to capture the compositionality of longer texts. One of the first projects towards a more sophisticated approach was proposed in [110]] where the author is using a recursive tensor neural network where the dependency parse tree of the sentence is used to learn word vectors in a bottom-up approach. The major drawback of this approach is that considers syntactic dependencies but cannot go beyond short sentences as it relies on parsing.

Another more sophisticated approach proposed by Mikolov [63] as an extension of word2vec that builds embeddings able to capture syntactic and semantic information from a complete text fragment (phrase, sentence, paragraph, text, ...) the Paragraph vector (PV) models or Doc2vec. Under the umbrella of doc2vec are two algorithms the Distributed Memory (PV-DM) and the Distributed Bag-of-Words (PV-DBOW). The extension is quite simple: insert one artificial word at the beginning of each text/fragment and apply the CBOW or Skip-gram techniques in the same way as before but considering the entire text/fragment as the context for the artificial word. In particular, a weight matrix containing the word embeddings W is used and share across all documents along with a paragraph embedding matrix D which learns the paragraph vector.

Before we proceed in the explanation of the models lets clarify what paragraph vector means. A " paragraph" is a text block with arbitrary length such as a section an actual paragraph or long sentence. In this section the word" paragraph" describes such text block of arbitrary length and the paragraph vector is like a word vector like as we mentioned above.

2.5.1 Distributed memory (PV-DM)

Distributed memory model is similar to CBOW but in paragraph vector framework, every paragraph Is mapped to a unique vector, represented by a Column in matrix D and every word is also mapped to a unique vector represented by a column in matrix W. So the fundamental difference is that, in this model a new matrix D is to learn and store the paragraph vectors. The paragraph vector, can be seen as another word but it acts as s memory that remembers what is missing from the current context.

The contents are fixed length and are chosen from a sliding window over the paragraph. However, the paragraph vector is not shared across all paragraphs but it is shared across all contexts generated from the same paragraph. On the contrary, the word vectors matrix W is shared across paragraph due to the assumption that the words have the same meaning for all paragraphs.

As in word2vec models, the paragraph vectors and words vectors are trained via stochastic gradient descent and the gradient is obtained via back prorogation. Thus in every step of stochastic gradient descent, a fixed length context is sampled randomly and the network computes the error gradient and then this gradient is used to update the parameters in the model as can been seen in the following figure 2.7

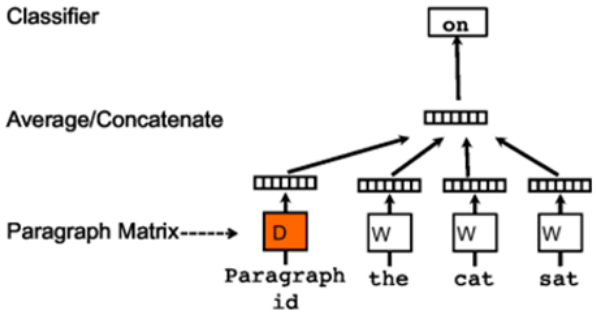


Figure 2.7: Distributed Memory model [63]

2.5.2 Distributed bag of words (PVDBOW)

All in all, a major advantage of paragraph vector is that can be learned from unlabeled data and thus can be useful in many tasks that not have enough labeled data. Additionally, paragraph vectors can obtain semantics of the words and preserve word order and thus deal efficiently with the key weaknesses of bag of words models.

The distributed memory model preserves the word order by concatenating the paragraph vector with the word vectors to predict the word in a text window. However, the authors, proposed an another algorithm that ignores the context word in the input but the model is forced to predict words randomly sampled from the paragraph in the output. Practically, that means in each iteration of stochastic gradient descent one random word is sampled from the text window and form a classification task given the paragraph vector as shown in figure2.8

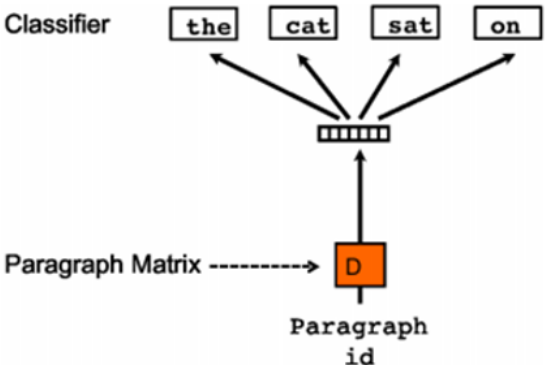


Figure 2.8: Distributed bag of words model [63]

Chapter 3

Feature learning

Feature engineering is important but labor intensive. It is therefore desirable to discover explanatory factors from the data and make the learning algorithms less dependent on extensive feature engineering. With the rapid growing of deep learning (representation learning[46]), many recent studies focus on learning the low-dimensional, dense, and real-valued vector as text features for sentiment analysis without any feature engineering. Existing deep learning methods for sentiment classification typically include two stages. In the first stage, they learn word embeddings from text corpus. In the second stage, word embeddings are applied to producing the representations of sentences/documents with semantic composition. Existing composition learning approaches are typically based on the principle of compositionality,which states that the meaning of a longer expression (e.g., a sentence or a document) comes from the meanings of its constituents and the rules used to combine them. Deep learning methods are combining feature extraction methods with classification. In the following paragraphs we describe basic principles of Neural networks and the reason of the quite recent success.

3.1 Neural networks

Neural networks are a machine learning technique incorporating principles that have been observed in the biological nervous system. A biological neuron is a cell consisting of a body, multiple smaller protrusions called dendrites and a single long protrusion called an axon. On top of the dendrites are found numerous spots, known as synapses, where axons of other neuros are creating a connection with the neuron. Often, axons are producing an electric pulse which affects the permeability of cell's surfuce which leads to the slight increase of the voltage inside the neuron body. The more activations come from other neurons, the higher the voltage grows.Given a certain threshold, if the voltage is below that threshold nothing particular happens. When this voltage exceeds these particular threshold the neuron produces an action potential meaning that a signal starts propagating through the axon. This axon can be attached to the dendrites of one or more other neurons, where the process is repeated in an analogous way.

This simplified function of biological neurons is stimulated by the artificial neuron networks meaning that the artificial neurons are units which contain one or more inputs and produce a single output. The activation of an artificial neuron depends on how much of non-zero inputs the neuron receives. It can remain either inactive and produce a zero output or it can produce an action potential which is represented by a non zero output. Multiple neurons are connected to one another, producing an (artificial) neural network.

Artificial neural networks (ANNs), also known as neural networks (NNs) are data-driven machine learning algorithms that process information by their dynamic state response to external inputs. [12]. Its origins are dated back to 1943, to a study of mathematical representations of information processing in biological systems by McCulloch and Pitts [113] In its most simple form a NN is a network of interconnected nodes and simple processing units known as neurons, the neurons are joined with weighted connections and calibrate the strength of the transmitted signals in a similar manner like synapses in human brain. Its resemblance with the brain is denoted by

two aspects; that the knowledge is acquired through the learning process and that it is stored in connections between neurons. [44]

Neural networks have been used in a wide range of machine learning tasks such as pattern classification or regression. The principal components of neural networks are neurons, layers, and activation functions despite the variety of the architectures that have been introduced over the years.

3.2 Perceptron

The fundamental unit of most neural networks is called a perceptron or simply neuron. Like its biological counterpart, a neuron can be seen as a sort of a "black box" which takes a fixed number of inputs and produces a single output. To each input, a weight is assigned indicating the extent to which this particular connection affecting the resulting output. The rough idea described above can be formalized as follows:

Let $X = (x_1, \ldots, x_n)$ represent the vector of input data and $w = (w_1, \ldots, w_n)$ be the vector of corresponding weights. In order the output to be calculated, we first need to compute the inner potential of neurons which is given as follows :

$$\xi = w_X = \sum_{i=1}^{n} w_i x_i \tag{3.1}$$

The activation functions are task at hand depended. In case of classification, as in our case, threshold activation functions are used. A threshold activation function in its primitive form can be given:

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq h \\ 0 & \xi < h \end{cases} \tag{3.2}$$

Where h is a fixed arbitrary real-valued parameter (threshold). Another form of artificial neuron is a neuron with bias. The bias neuron has an extra input x_0 and a corresponding weight w_0 are considered in order the weighted to be computed. If the $w_0 = -h$ then the activation function takes the following form:

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 0 \\ 0 & \xi < 0 \end{cases} \tag{3.3}$$

Where ξ now starts from zero instead of one. This approach is the dominant approach in neural network literature henceforth any mention of term "neuron" is only to been as a neuron with bias.

3.3 Activation

The output of each node is calculated by the node's activation function that takes weighted inputs of the node as parameters transformed from a transfer function. The transfer function aims to create a linear combination of weighted inputs in order to provide them to the activation function. In the sections that follow briefly, we discuss activation function that we are going to use later in this thesis

3.3.1 Logistic or Sigmoid function

A sigmoid function is a monotonically increasing function, which reached an asymptote at some finite value as $\pm\infty$ is reached. In terms of neural networks, the most widely used sigmoid function is the standard logistic defined as:

$$\sigma(\chi) = \frac{1}{1 + e^{-x})} \tag{3.4}$$

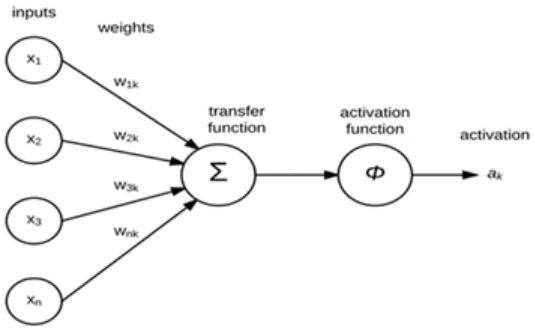


Figure 3.1: The role of the activation function in the neural network model

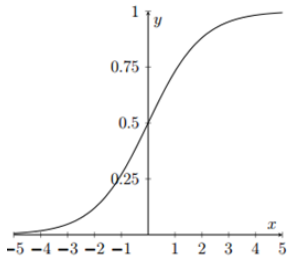


Figure 3.2: Standard Sigmoid function

The logistic or sigmoid function is a smoother approximation of the step function that was used in the early versions of neural networks. The output values are in range $[0, 1]$ hence it is suitable for output neurons that perform classification task. However, an important problem arises, when the weights of the network are initialized with small values almost zero. The initial activations for the standard logistic function will be then set 0.5 on average. Thus, sigmoid that are symmetric about the origin are preferred to be used because tackle this problem by producing always positive outputs and help gradient-based optimization. From optimization point of view, another problem occurs. The problem is that the derivatives for sigmoid are vanishing near saturation points, Thus, this makes it harder for a neuron to propagate the error signal and move out from the saturation points.

3.3.2 Hyperbolic tangent (tanh)

The Hyperbolic tangent is a non-linear S-shaped function like the sigmoid mentioned before but its lower horizontal asymptote is at -1 instead of 0. The fundamental difference between those two is that the tanh is zero-centered with a steeper rise which leads the classification models to reduce the number of misclassified samples. The tanh functions are differentiable and is given:

$$tanh(x) = \frac{e^{\chi} - e^{-x}}{e^{\chi} + e^{-x}} \tag{3.5}$$

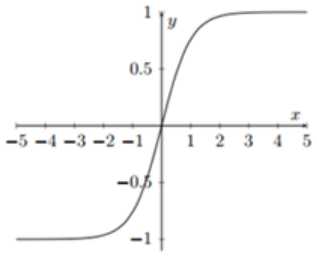


Figure 3.3: Standard hyperbolic tangent function

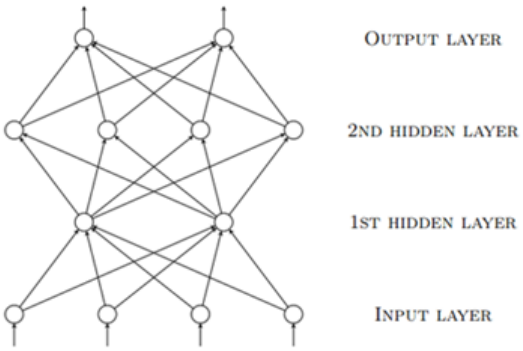


Figure 3.4: An example MLP network with two hidden layers

3.3.3 Rectified linear unit (ReLu)

The rectifier activation function has been shown that can improve the discriminative performance of convolutional networks[33]. The rectifier nonlinearity is defined as follows:

$$ReLU(x) = max(0, x) \tag{3.6}$$

It has been shown that ReLu gives true sparsity to the model due to zero constrain, which means that only a small set of values are nonzero. Moreover, it does not present vanishing gradient problem because the function is becoming linear when $x > 0$ Additionally, sparse representation is biologically plausible which means that for a given set of neurons, most of them are inactive and just a few are activated by some input. The main reason that makes sparsity desirable to machine learning algorithms is that sparse features are effectively lower the number of dimensions and sufficiently tackles the so called curse of dimensionality. Thus, sparse representations are more robust in small changes to the inputs compared to dense representations. It has been proven that the discontinuity at zero can affect optimization techniques. However, when smoother versions of rectifier were applied the performance was worse due to the fact that that exact sparsity was lost.

3.3.4 Softmax

The Softmax activation function is widely used in the last layer of the networks, it aims to interpreter an arbitrary real value to the posterior probability of the class in range $(0, 1)$:

$$p(c_k \,|\, x) = \frac{e^{ak}}{\sum_{i=1}^{m} e^{ai}} \tag{3.7}$$

Where m is the number of output neurons (classes) a_k is the activation value of the $k-th$ node:

$$a_i = \sum_{j=0}^{d} w_{ij} h_j(x) \tag{3.8}$$

Where w_ij is the $i-th$ node's weights and the $h_j(x)$ is the output of the previous layer.

3.4 Multi-Layer perceptron

In case we have only one neuron unit, the output value is used as the final indicator of the assigned class to the input vector. Someone, could feed this output to another neuron instead. Hence, in that way it is possible a network consisted of interconnected neuros to be created. Numerous different topologies have been proposed from different perspectives and different purposes. However, the most widely used topology of neural networks are the multi-layer feedforward networks. Since these have been shown state of the art results in sentiment analysis are of particular importance of this thesis.

Having explained some baseline information about neural networks now we are about to introduce formally the simplest version of neural network, the multi-layer perceptron (MLP) ([103],[127]). The network consists of a certain number of neurons that are then split into several disjoint sets, the so called layers. The layers in MLP architecture are ordered sequentially and each layer receives input from the preceding layer. The input layer is not considered as a "true" layer because no computation is performed in it. It receives problem-specific inputs from the outside world. An MLP contains one or more hidden layers, which receive inputs from preceding layers (input or hidden layers) and their outputs connect to the next layers (output or hidden layers). Each neuron in a hidden layer employs a nonlinear activation function that is differentiable. The output layer presents the final result of the computation performed by the network to the outside world. For a given input layer x, the network computes the hidden activation vector h and the output vector y as follows:

$$h = f(W^x hx + b^h) \tag{3.9}$$

$$y = G(W^x yh + b^y) \tag{3.10}$$

Where W are the weight matrices of two connected layers, W_{xh} is the matrix that contains the weights of input layer, Why is the matrix that contains weights of hidden layer , b are the bias vectors and F and G are the activation functions.

The network is highly connected thus all the neurons in one layer are connected to all neurons in the following layer. Hence, multi-layer perceptron is a feed-forward NN which means that the information inside the network moves from input neurons through hidden layers' neuros to the output neurons.

The MLP's expressive power is given by the universal approximation theorem [34]. The theorem states that a single hidden layer MLP and sufficient number of nonlinear units are sufficient to approximate any smooth function and a compact input domain with arbitrage precision. However, the number of hidden units that are required is unknown and sometimes can be so large and thus inapplicable. The use of hidden layers helps the partition of input space into exponentially more linear regions than a shallow network, with same number neurons does. That ability allows them to represent easily highly structured and complex functions. MLPs have exhibited excellent results in pattern recognition and more specifically in natural language processing and sentiment analysis.

3.5 Training

3.5.1 Loss function

The optimization objective for supervised neural network training is based on a loss function that measures how well the network performs on the training data. As the name implies, the output of the loss function needs to be minimized to receive the best-performing model (as opposed to maximization of the likelihood of the data under the model as it is common for MaxEnt). There are multiple loss functions which are common in the literature, and which are usually preferably used with specific output activation functions. We will briefly introduce the two most common functions. In the following, let $\hat{y}$ be the target value and y the predicted output.N denotes the number of examples, D the output dimensionality, i.e., the number of units.

Mean squared error (MSE) The MSE function assumes that the errors are normally distributed. It is defined as:

$$MSE(y, y) = \frac{1}{N} \sum_{n=1}^{N} (y_i - y_i)^2 \tag{3.11}$$

MSE assumes that the errors are normally distributed. It is frequently used in combination with the tanh and linear activation functions.

Cross-entropy error (CEE) The CEE function measures the cross-entropy of the output and the target values. As a consequence, it is only applicable if the predictions and targets are probabilities. It is a natural choice for sigmoid and Softmax activation functions. When used in combination with

one of these functions, the computation of the derivatives in backpropagation can be simplified significantly. The CEE function for the multiclass case, i.e., when using Softmax activation, is defined as follows:

$$CEE(Y, Y) = \sum_{n=1}^{N} \sum_{d=1}^{D} \hat{y_{nd}} ln\, y_{nd} \tag{3.12}$$

3.5.2 Backpropagation

The concept of backpropagation algorithm was first introduced by Paul Werbos in his 1974 PhD thesis. Backpropagation is a neural network training strategy. In supervised learning. Target classes are essential the baseline for error calculation. Then errors are backpropagated to each node in previous layers. Error e is obtained as gradient of the loss function L with respect to each layer's weights w_kj given input of the node x and an activation function.

$$a_j = \varphi(\sum_{k=1}^{n} w_k j x_k) \tag{3.13}$$

$$e = \frac{\partial L}{\partial w_k j} = \frac{\partial L}{\partial w_k j} \frac{\partial a_j}{\partial w_k j} \tag{3.14}$$

Gradient computation demands application of the chain rule in order to compute partial derivative of the loss function L with respect to particular weight wkj . Using the error, weights are updated by an optimization algorithm such as stochastic gradient descent. Basic stochastic gradient it usually leads to slow convergence of the network. Thus, several techniques have been proposed in literature for optimization algorithms such as Adagrad [28], Adatadelta [134], RMSprop, and Adam [58] all of the which can improve greatly the speed convergence. Here we briefly describe RMSprop and Adam which will be used in the subsequent sections of this work.

3.5.3 Optimization algorithms

RMSprop RMSprop is an optimizer having per-parameters adaptive learning rate. Incorporates a moving average of the magnitude of recent gradients in order to normalize the current gradients. The normalization is performed over the root mean soared gradients. The learning rate or the step rate and the running average term $r(\tau)$ is added to the weight update equation:

$$r(\tau) = \gamma r(\tau - 1) + (1 - \gamma)(\frac{\partial E}{\partial w_i})^2 \tag{3.15}$$

$$\Delta w(\tau + 1) = \frac{-\eta}{\sqrt{r(\tau)}} * \frac{\partial E}{\partial w_i} \tag{3.16}$$

Where γ is the decay value which calibrates the contribution of new gradients for the running average $r(\tau)$ One key aspect is to initialize weights W and biases b to positive and negative values close to zero, in order the sigmoidal activation functions to operate on the central region which leads to larger propagated gradients [32]. Usually, the values are drawn in a random and independent order from uniform or Gaussian distributions. The inputs features, for the reason we mentioned above, are normalized so as to have zero mean and unit variance in each dimension.

Adaptive Moment Estimation (Adam) Adam [58] is the second method that we examine. This method computes the adaptive learning rates for each parameter. But in contrast of RMSprop, Adam is not only keeps an exponentially decay average of the past squared gradients u_t but it keeps also an exponentially decaying average of the past gradients m_t :

$$m_t = \beta_1 m_(t - 1) + (1 - \beta_1) g_t \tag{3.17}$$

$$u_t = b_2 u_(t - 1) + (1 - \beta_2)(g_t)^2 \tag{3.18}$$

$m_t, u_t$are the estimates gradients of first and second moment respectively.
The Adam update rule is given as follows:

$$\theta_{\tau+)} = \theta_\tau - \frac{h}{\sqrt{\hat{u_t}} + \varepsilon}\hat{m_t} \tag{3.19}$$

Usually the default values are 0.9 for β_1 10-8 for ε and 0.999 for β_2

3.6 Deep learning

As we mentioned above, despite the fact that neural networks exist for almost six decays their great successes are quite recent [39, 60]. This great success is due to two factors; technological and theoretical. On the technological point of view, faster computers, and the use of graphical processing unit (GPU) instead of till now used CPU made possible to train large networks with millions of parameters in reasonable time. Secondly, from the theoretical point of view, the breakthrough was the discovery that the use of multiple hidden layers leads to exponentially grow of the expressiveness of the models ([42],[8],[80]). Moreover, the use of ReLu, dropout layers and proper random initialization techniques can be seen as significant improvements in theoretical point of view.

The typical approach in classic machine learning is to engineer features by hand and then to feed them in a general purpose classifier such as support vector machine (SVM) [106]. Although this method was widely used in early stages of machine learning for complex task such as natural language processing does not holds because it is hard to engineer features that are sufficiently expressive. Additionally, when the number of features and thus dimensions grows exponentially, known as the curse of dimensionality, the data are becoming sparse. Without the use of distributed representations, it is not trivial to generalize to regions of the space that have no training instances which leads to poor classifiers generalization. ([9],[5])

Deep learning arises from this need. A deep model is usually comprised of several hidden layers of computations that are used to automatically discover more complex features and allow their composition. By learning and combining multiple levels of representations, the number of distinguishable regions in a deep architecture grows almost exponentially with the number of parameters, with the potential to generalize to non-local regions unseen in training. [8] Deep learning models are undoubtedly powerful models that take advantage of the learning process as described above. Deep learning models use backpropagation to compute the gradients in order to be trained and updating the weights so as to minimize the cost function on the training set. It is not guaranteed that the global minimum will be achieved, but has been argued that in high dimensional spaces the majority of local minimum errors approximate the global minim error and that sable points are responsible for the learning stalls.([91],[16])

3.6.1 Feedforward neural networks

A common type of neural networks is the feed-forward type of networks. As the name hints, in feed-forward neural networks the information flows only forward through them. This architecture is among the simplest possible networks and one of the first that was used . They are comprised of layers of neurons, where neurons in one layer are connected to some or all in the next. The first layer is called the input layer, the last is the output layer and the layers in between are the hidden layers. The simplest form of feed forward network is a fully connected one, where each neuron in one layer is connected to all neurons in the next one.

Similar to general neural network information flows between neurons by weighted sums of the connected neurons followed by an activation function. n. Given a weight matrix $W^l$where $W_.(i,k)$is the weight from neuron k in layer$l-1$ to neuron i in layer l, the biases $b^l(i)$ and a layer of hidden neurons h^l, the value for the $i'th$ neuron $h^l(i)$ in layer l is calculated as shown in equation. Note that we use the logistic function as the activation function.

$$h^l(i) = \sigma(b(i) + \sum_k W^l(i,k) * h^{l-1}(k)) \tag{3.20}$$

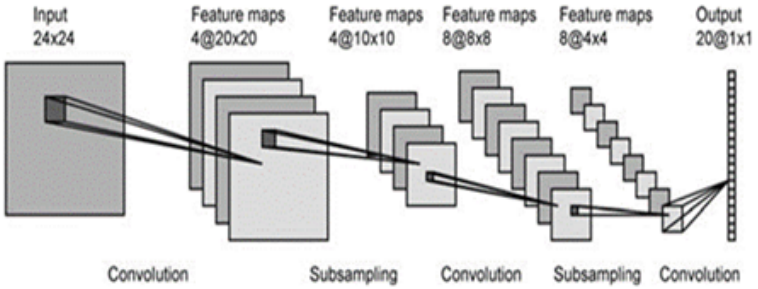


Figure 3.5: A Simple CNN Topology

Taking all neurons of a layer into account, the neurons in the next layer h^l is calculated with the matrix multiplication in equation 3.20. Recall that h^l*is* a vector of neurons, thus $\sigma(.)$ will be applied element-wise on its argument

$$h^l = \sigma(W^l h^{(l-1)} + b^l) \tag{3.21}$$

The input layer has no connection to it, instead its values are set manually each time the network is run. Likewise, the output layer has no connections from it, since its values are taken as the result of the network's computations.In literature, are many from of feedforward neural networks, two of the most notorious in classification task are the multi-layer perceptron (MLP) and the convolutional neural networks (CNNS).

3.6.2 Convolutional neural network

Convolutional Neural Network are feedforward neural networks that are composed of one or more convolutional neural layers (often with a subsampling step) followed by one or more fully connected layers as in a standard multilayer neural network. The result of the convolution is then passed through a non-linear activation function, typically a ReLU. Other hidden layers often can be seen in a CNN structure are the pooling layers. Pooling layers are responsible for downsampling the output of the convolution layers using the rule of the mean or maxim value of the region. This leads to the reduction of overfitting and provides some translation invariance. The output of the CNN is a vector whose values are interpreted as the conditional probabilities of each class input. A depiction of a CNN is presented in Figure 3.5

Originally the convolutional neural networks were invented for computer vision but have shown to be effective in numerous other fields such as natural language processing. The convolution process is responsible for the generation of a feature map can formally defined as follows:

$$o[n] = f\,[n] * g\,[n] = \sum_{u=-\infty}^{\infty} f\,[u]\,g[n-u] \tag{3.22}$$

$$= \sum_{u=-\infty}^{\infty} f[n-u]g[u] \tag{3.23}$$

In 2- Dimensional Space:

$$o[m,n] = f[m,n] * g[m,n] = \sum_{u=-\infty}^{\infty}\sum_{u=-\infty}^{\infty} f[u,v]g[m-u,n-v] \tag{3.24}$$

$$= \textstyle\sum_{u=-\infty}^{\infty}\sum_{u=-\infty}^{\infty} f[m-u,n-v]g[u,v]$$

where o is the output function; f *and* g are the input functions. In subsampling process the dimensionality of feature maps is reduced. As we mentioned there are several types of layers of which a CNN can be comprised:

Convolution Layer: A convolutional layer consist of K filters. Loosely putted, its input has one or more feature maps. The output of the layer is K feature maps, each of whom is computed as the convolution of the input and a filter k, plus its bias:

$$h_i j_k = \varphi((W_k * x)_i j + b_k) \tag{3.25}$$

Where i and j are the row and column index respectively, φ is layer's activation function and x its input. Thus the output of a convolutional layers is the same dimension as the input and its depth is defined by the number of filters.

Pooling layer: A pooling layer is usually placed between two successive convolutional layers in a CNN. It aims to downsamples the input matrix, thus its reducing the space representation and the number of feature parameters. Although the depth dimension it remains unchanged. Usually a pooling layer divides the input into nor-overlapping rectangle regions the size of whom is defined by the pool shape. Then, it produces the output values for each region by using max sum or average operator. Thus if a pooling layer uses the max operator, it is called a max pooling layer. The pool size is normally set as (2; 2) as larger sizes may lost too much information.

Fully-connected layer The reason of the existence of fully connected layers placed at the end of a CCN is to refine features learned from the convolution layers or to return class scores in the classification task which is our case.

3.6.3 Recurrent neural network

Feedback connections in a neural network can produce past context information. This network architecture is known as recurrent neural network (RNN). Many versions of recurrent neural networks have been developed or adapted in order to achieve excellent results in different machine learning domains.

For a sequence of input vectors $x_1 \ldots, x_T$ a basic RNN computes the sequence of hidden activations $h_1, \ldots, h_T$ and the output vectors as $y_1, \ldots, y_T$ as:

$$h_t = f(W^{hx}{}_t + W^{hh} h_{t-1} + b^h) \tag{3.26}$$

$$y_t = G(W^{hy} h_t + b^y) \tag{3.27}$$

For all times steps $t = 1, \ldots, T$ where W are the weights matrices of two connected layers , and B denotes the bias terms as F and G are the activation function used. For a deep RNN with several stacked hidden layers, each hidden layer receives the output of the previous hidden layer. Despite this minor modification, the effects are profound: In a RNN information obtained from previous time steps can loosely speaking circulate indefinitely inside the network through the directed cycles, where hidden layer can play the role of memory. These hidden activations are making an internal state where can be represented as h_t vector given a certain time step for each hidden layer. The output of the network at time t is a function of all received inputs vectors till that time. This RNN as the we one we described is also known as a vanilla RNN.

From the scope of memory, RNNs are more computationally expensive than FNNs. As we stated above, a feed forward neural network can approximate any non-linear function on a compact domain with arbitrage precision. The equivalent to MLPs approximation theorem in RNNs is that with sufficient number of hidden units any dynamical system can be approximated. Potentially a RNN is computationally as expressive as any Turing machine [17],[40]

In order the RNN to be trained a straightforward extension of the backpropagation algorithm is applied known as backpropagation through time (BPTT) [109]. The basic idea behind this training strategy is to unfold the network through time and then to use the standard backpropagation algorithm as it were a classic MLP. This unfolded presentation denotes that a RNN can be else seen as a very deep FNN but with a layer for time step and share weights across time.

The above vanilla structure is not commonly used because of a problem known as vanishing gradient problem and a subsequent problem called exploding gradient problem [8, 47]. Several

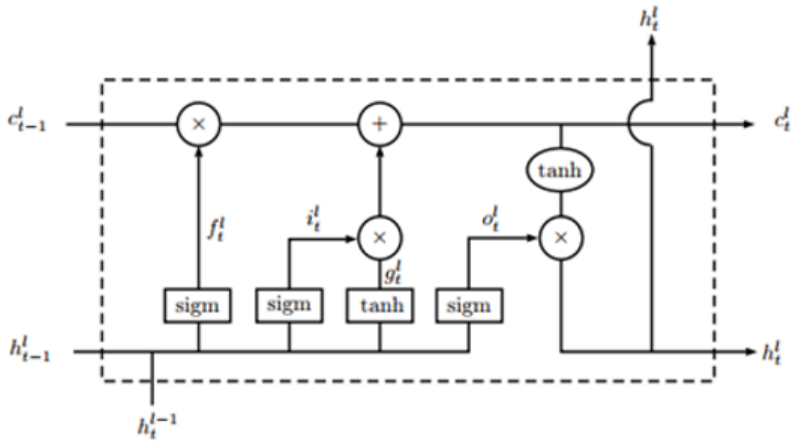


Figure 3.6: Basic LSTM memory block

techniques have been presented to overcome the difficulties of training RNNs, such as training with second order optimization methods (e.g. Hessian-free [74]), initialization of the recurrent weights with scaled identity matrix (IRNN) [64], unsupervised pre-training using restricted Boltzmann machines (RBMs) [11], linear autoencoder network initialization [90], and more complex architectures such as the long short-term memory.

3.6.4 Long short-term memory

The RNN owes its memory to the fact that it feeds its stated forward in time, in practice the network easily forget after a couple of steps. It has been shown that to train a standard RNN to find patterns that occurs after a large number of steps is a trivial task[6]. In order this problem to be addressed an extension of standard RNN was proposed, the so-called Long Short-Term Memory (LSTM).

An LSTM network architecture is built upon LSTM memory blocks just as a RNN is based on McCulloch-Pitts neurons. LSTM as almost all feedforward neural networks is comprised of multiple layers with many cells in each layer. Figure 3.6 shows a basic LSTM memory block. This is a single cell k in layer l at time step t. As we can see the output is a single entry in the input vector to the next layer and next time step. There have been several proposed architectures, we choose to describe here the model as presented in its first form in 1997 [48] .

The rough idea behind the LSTM cell is the use of the so called gates to control which information the cell will remember, forget and produce. These gates allow cell information longer that the neuron of an RNN. In time step t the output of l is defined as ${h_t}^l$. Similarly, for the LSTM memory cells ${c_t}^l$ defines the memory states for layer l at time step t. Formally the calculation of new memory state is as follows:

$$ {c_t}^l = {f_t}^l \odot {c_{(t-1)}}^l + {i_t}^l \odot {g_t}^l \tag{3.28} $$

Where $\odot$ denotes element-wise multiplication. Furthermore, the previous memory state is element-wise multiplied by f_t, the forget mechanism. The forget mechanism is responsibly to choose which properties of c_{t-1}^l) and is given as follows:

$$ {f_t}^l = \sigma \left[\mathrm{T}_{(m+n,n)} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} + b_f \right] \tag{3.29} $$

The size of layer $l-1$ and l is m and n respectively, thus the transformation T is done on a vector of size $m+n$ into a vector of size n. The use of element wise logistic function secures that the values are between 0 and 1 where zero denotes that the cell must completely forget and 1 to remember what in time t is stored in the memory state. The additional bias term is used so as the model to store information easily during the first steps of training [?].

When the information from input and the previous output ${i_t}^l \odot {g_t}^l$ is added to the memory gate then the gate decides which features g_t^l of each input should be added. The values of i_t^l, g_t^l are given from the following Equations:

$$ {i_t}^l = \sigma \left[T_{m+n,n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \right] \tag{3.30} $$

$$ g_t^l = tanh \left[T_{m+n,n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \right] \tag{3.31} $$

Note that g l t use the tanh function instead of the logistic function. This allows the input values to the LSTM to take on values between -1 and 1.

After the memory state is updated, it needs to be decided what the cell should output. This is done with the last gate o_t, which regulates what properties of our memory state we will use in the output and is described in the following equation:

$$ o_t^l = \sigma \left[T_{m+n,n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \right] \tag{3.32} $$

The output for the current layer h_t^l is sent to the next layer and time step. The tanh function is first applied to the memory state to normalize it to the interval $[-1, 1]$ and the result is then scaled with the output gate.

$$ {h_t}^l = {o_t}^l * tanh({c_t}^l) \tag{3.33} $$

In equation 3.34, lstm(.) is denoted as a function calculating the next output of an LSTM-cell according to equations 3.28 , 3.33 (The internal memory state is implicitly updated).

$$ {h_t}^l = lstm_t^l({h_1}^{(l-1)}, {h_{(t-1)}}^l) \tag{3.34} $$

3.6.5 Bidirectional RNN and LSTM

In many tasks involving classification over temporally or spatially sequential inputs, we might be interested in looking also at future samples before outputting a label for the current timestep. For example, when classifying a handwritten letter inside a word it is beneficial to know the following letters as well as the preceding ones. Using a time-window to incorporate future samples, poses strong limitations due, for example, to the fixed length of lookahead. Delaying the output by n timesteps would still impose a fixed lookahead and it would force the network to remember the original input— appeared n steps before—and the previous context throughout the delay. [36]

Bidirectional recurrent neural networks (BRNNs), introduced in [107], are a clever solution to the problem. In a BRNN each hidden layer is split into two separate layers, one reads the training sequences forwards $(x_1, ..., x_T)$ and the other one backwards $(x_T, ..., x_1)$. The sequence is fully read and the hidden activations are stored for all timesteps in each of the two distinct layers. Finally, the computed activations are fed to the next layer, giving the network full and symmetrical context for both past and future samples of the input sequence. By substituting simple recurrent neurons in a BRNN with LSTM units we obtain a bidirectional long short term memory (BLSTM) network as shown in figure 3.7.

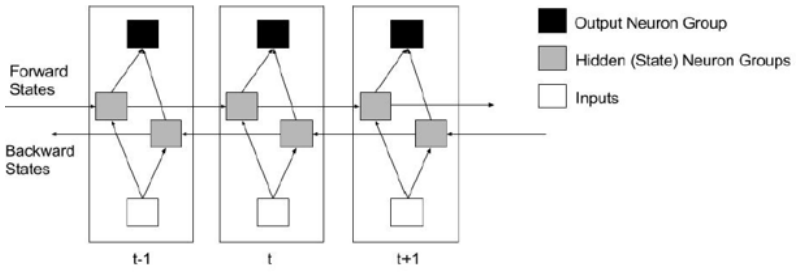


Figure 3.7: A bidirectional long short term memory (BLSTM) network

The violation of causality is not a concern in tasks where at test time the input sequence is completely available. This is the case of spatially sequential inputs (such as images), or temporally sequential inputs when the data was fully acquired before. Even data from a short amount of time ahead is sufficient in many tasks that need the output at the end of an input portion, such as in automatic speech recognition, where outputs are only required at the end of a word.

RNNs using LSTM and BLSTM units have advanced state-of-the-art results in many challenging tasks such as speech recognition [39], machine translation [114], unconstrained handwriting transcription [38]and generation [37], image captioning [123], language modeling [133].

Chapter 4

Classification

Classification can be expressed as a categorization process where objects are recognized, differentiated and understood. A data object represents an entity, typically described by attributes. Data objects become data tuples when stored in a database, where rows correspond to the attributes. Classification process finds a model that describes (discrete, unordered) data class labels. This model is made of on the analysis of a set of training data. Purpose of this model is to predict the class label of objects for which the class label is unknown. Classification is usually referred in machine learning to supervised learning. There are also unsupervised, semi-supervised and active learning methods available to perform classification. Many classification methods have been proposed by researchers in machine learning, pattern recognition, and statics. A brief description of machine learning is given followed by the presentation of the classifiers used in this thesis.

4.1 Machine learning

Machine Learning (ML) is a sub-field of Artificial Intelligence whose objective is to develop, understand and evaluate algorithms and techniques that allow a computer to learn. Machine learning incorporates techniques from statistics, human psychology, and brain modeling. Human psychology and neural models obtained from brain modeling help in understanding the workings of the human brain, and especially its learning process, which can be used in the formulation of ML algorithms.

The machine learning algorithms are mainly based on statistics, and both of them sharing many concepts with the same functionality but come with a different name. For instance, inference in statistics is the task that going from particular observations to general descriptions whereas in machine learning estimations is the learning process and discriminants is the classification task.

During the first steps of machine learning, it was believed that there is a need of a new type of thinking or new models to be introduced in order machine learning algorithms to be highly accurate. However, the recent success of machine learning algorithms pointed out that we actually need huge amount of learning data in order the algorithms to produce remarkable results. Machine learning algorithms found use in extremely wide variety of tasks from sentiment analysis to stock price prediction.

Formally, the objective of machine learning is to find a function $f : X \Rightarrow Y$ which interpreters the data (X) to prediction (Y). The function belongs to a certain function class, which consists of different learning algorithms. The elements of X and Y are applications-specific. The rough idea behind the learning process can be seen as in Fig.4.1 .

The input data or the train set consists of observations, memory storages and factual basis for further reasoning. In the abstraction phase the data are translated into general representations and give them a meaning. Abstracted connections are a basis of knowledge representation. In knowledge representation, the data inputs are summarized into a model, that is explicit description of the structured patterns among data. The process, where a suitable model is shaped, is called training [3]. The model, does not aim to replicate train data but to predict new unseen samples. The ability of a model to predict new instances is call generalization. Generalization uses abstracted data to perform action and makes the model useful for applications. While every possible model

Figure 4.1: Machine learning process

could be examined, it is not feasible. Thus, learners use heuristics to reduce the possible models. With heuristics, there comes also a bias towards certain solutions. The learning process is a trivial task due a number of factors.

There several issues to be addressed concerning the learning processes itself. To begin with, in learning the problems are called ill-posed. An ill posed problem, has no unique solution and the solutions are not optimal. Therefore, different solutions with different training data in the same problem could presented. Thus with more training data available a problem becomes less ill-posed.

Moreover, in order a problem to have a unique solution, additional assumptions should take place the so-called inductive bias. For example, choosing an optimizing criterion (an error to be minimized) is an inductive bias. An inductive bias for the data, which the learning algorithms often use, is independent and identically distributed samples. These kind of samples are all drawn from the same joint distribution.

However, the independent and identically distributed assumption might not hold because usually the distributions are not identical. In order this problem to be addressed, machine learning uses the covariate shift assumption. In other words, in machine learning is assumed that the training set and the test set follow different probability distributions but the distributions of the output predictions in respect of input values are unchanged.

A fundamental problem, that one should address in machine learning is when the model underfits or overfit. A model we say that is underfits when the complexity of the model is less than the complexity of the function that generates the data. In other words, fitting a linear model to the data produced by a general polynomial function will lead to model underfitting. On the other hand, when the model exhibits perfect results on training set but fails to generalize the model then we say that is overfitting.

Last but not least, a major issue in machine learning is the huge number of dimensions. The so called curse of dimensionality affects the time and space complexities leading in slower and computationally expensive algorithms. In order this problem to be addressed several dimensionality reduction techniques have been proposed in literature.

The dimensionality reduction techniques fall into two categories: feature selection and feature extraction. In feature selection the k most informative features are picked while the other features are being discarded. In feature extraction, a new set of k dimensions are found, that are combinations of the original dimensions. A widely used method for feature extraction is called principal component analysis which was used in this thesis.

4.2 Machine learning techniques classification

A common categorization of machine learning methods splits them into supervised, unsupervised and reinforcement learning algorithms and can be found in many books of the field.

4.2.1 Supervised learning

Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier (if the output is discrete, see classification) or a regression function (if the output is continuous, see regression). The inferred function should

predict the correct output value for any valid input object. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

The first step is to use pairs of the training set to teach the algorithm some correct mappings of the problem space. This is where the “supervision” comes from, the learner is told what the right outcome for the function is for an example and is therefore supervised. Afterwards the algorithm can be confronted with unseen inputs/samples for which it determines the outputs/target, solely based on its experience with the training set. The algorithm is said to generalize over the problem space.

Besides the classification of inputs, another typical ML task is regression. Regression comes very close to interpolation, which is the task of finding a function that is exactly correct in the points given by the training set. Regression adds the assumption that the samples of the training set are affected by noise. Thus, the task is to find an approximation for the function, from which the training set was sampled. It turns out that the tasks of classification and regression basically solve the same problem and the solution of one can also be used for the other using an appropriate transformation. Most of the widely known ML techniques belong to this category, such as Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Bayesian Statistics, Kernel Estimators and Decision Trees.

Text Classification problem

Classification is about deciding which class a certain data item belongs to. A machine learning system that does the classification is called a classifier. The classifier learns a discriminant, which is a function that separates the data items to different classes. Thus, the classifier can predict in which class a new data item belongs to by using the discriminant

Loosely speaking, the problem of text classification can be formulated as follows: in Given a description $d \in X$ where X is the document space, and a fixed set of categories, which in our case is two C= {positive, negative} and a labeled train set consist of documents $(d, c) \in X * C$ using a machine learning algorithm Γ so as to train a classifier or classification function $\Gamma(D) = \gamma$ in order to map the document to classes: γ : X$\rightarrow C$

4.2.2 Unsupervised Learning

In unsupervised learning the machine simply receives inputs x_1, x_2 but obtains neither supervised target outputs, nor rewards from its environment. It may seem somewhat mysterious to imagine what the machine could possibly learn given that it doesn't get any feedback from its environment. However, it is possible to develop of formal framework for unsupervised learning based on the notion that the machine's goal is to build representations of the input that can be used for decision making, predicting future inputs, efficiently communicating the inputs to another machine, etc. In a sense, unsupervised learning can be thought of as finding patterns in the data above and beyond what would be considered pure unstructured noise [31].

A typical example application would be the recommendation system of an online shop. It analyzes which products should be advertised to a customer based on information about products he bought before and products bought by other customers purchasing the same articles beforehand. It is closely related to density estimation in the field of statistics. Typical approaches are clustering such as with the k-Nearest Neighbor Algorithm, Hidden Markov Models (HMMs) or the self-organization of ANNs.

4.2.3 Reinforcement learning

In reinforcement learning, the machine interacts with its environment by producing actions $(a_1, a_2...)$ These actions affect the state of the environment, which in turn results in the machine receiving some scalar rewards (or punishments) $r_1, r_2, ...$ The goal of the machine is to learn to act in a way that maximizes the future rewards it receives (or minimizes the punishments) over its lifetime [31]. Reinforcement learning is closely related to the fields of decision theory (in statistics and management science), and control theory (in engineering).The fundamental problems studied in these fields are often formally equivalent, and the solutions are the same, although different aspects of problem and solution are usually emphasized.

4.3 Machine learning methods

In the following paragraphs , we present the Machine learning classifiers we are going later to apply in order to tackle the problem of sentiment Analysis.

4.3.1 Naïve Bayes

In [125] a distinguish of two naive Bayes models are presented [20] the multinomial and the Bernoulli. In the multinomial naïve Bayes model, a term is generated from the vocabulary in each position of the document. On the other hand, the Bernoulli model generates an indicator for each term of the dataset. If the values are 1 that indicates that the term is present and 0 is used to indicate absence. This is similar to unigram model that we discussed in section 2. Moreover, the Bernoulli model does not take into account multiple occurrence as multinomial naïve Bayes does. In this thesis multinomial Naïve Bayes is considered. The Bayes theorem 4.1. is used widely in machine learning techniques, but in its simplest form can be found in naïve Bayes classifier. The Bayes theorem states that given two random variables X and Y:

$$P(Y\,|\,X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{4.1}$$

The theorem provides a strict mathematical rule explaining how you should change your existing beliefs based of new evidence. In fact, it allows to calculate unknown conditional probabilities from a known conditional probability along with the prior probabilities. The theorem is based on the assumption that the variables are in depended from each other within each class. In other words, the presence of a feature is not correlated with the presence of any other feature. This is formalized in equation 4.1. Although the conditional independence assumption does not hold the model surprisingly performs quite well. Equation 4.3. states the model, where C denotes the class label,$F_1, \ldots, F_n$ are the features . Since the denominator of equation 4.1 does not depend on C, that makes it a constant scaling factor $\frac{1}{z}$ for equation 4.3. In a more mathematical expression it would be: the posterior probability is computed by multiplying the evidence scaling factor with the prior probability, multiplied by the product of the independent likelihoods. The main advantage of the use of Naïve Bayes Model is that can be trained in a small training set because independent variables are assumed only the variables for each class. Additionally, the performance is quite good and in this thesis the model is used as a baseline for other models.

$$P(X_i\,|\,Y, X_j) = P(X_i\,|\,Y),\ for\, i \neq j \tag{4.2}$$

$$P(Y|F_1, \ldots, F_n) \propto \frac{1}{z} P(Y) \prod_{i=1}^{n} P(F_i|C) \tag{4.3}$$

4.3.2 Multinomial Naïve Bayes

The multinomial Naïve Bayes is a pure probabilistic method [18]. The probability of each document d of being in class c is given:

$$P(c\,|\,d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k\,|\,c) \tag{4.4}$$

Where $P(t_k\,|\,c)$ is the conditional probability of t_k occurring in a document of class c. $P(c)$ is a prior probability of a document occurring in class c. If the conditional probability of a term does not implicitly be categorized in one class, the category with higher probability is then chosen.

The utter goal of the classification is obviously to find the best class for each document. The best class for Naïve Bayes classification algorithm is the most likely or maximum a posterior class c_{map}

$$c_{map} = argmax_{c \in C} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c) \tag{4.5}$$

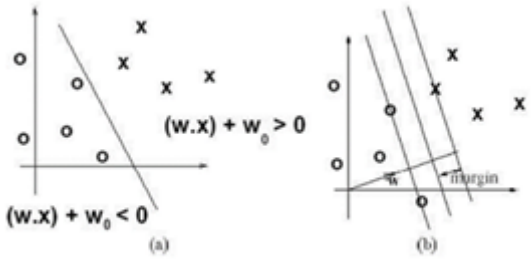


Figure 4.2: SVM Margin

In Equation 4.5 $\hat{P}$ are true values of the parameters $P(c)$ and $P(t_k|c)$ are the unknown but they can be calculated from the training set. As can be seen many conditionals probabilities are multiplied which leads to a floating point underflow. In order this problem to be addressed summing logarithms of probabilities were proposed. The class with the highest log probability is still considered to be the best class. The logarithm function is monotonic and hence the maximization of such a function is performed in most Naïve Bayes variations is:

$$c_{map} = argmax_{c \in C} \hat{P}(c) \sum_{1 \leq k \leq n_d} logP(t_k|c) \tag{4.6}$$

The conditional parameter $P(t_k|c)$ is used as weight in order the indicator t_k to be evaluated if is representative or not for the class c. Similarly, the $logP(c)$ is a weight that indicates the relative frequency of class c. Therefore, the sum of log prior and the term weights indicates which class is suitable for each document.

4.3.3 Support vector machine

Support vector machines were first introduced in 1995 by Cortes and Vapnik [22]. The main objective of SVMs is to project the datapoints, belongs to a two class train set, in a higher dimensional space where a maximum margin separating hyperplane between the datapoints of two classes need to be found.

The input of SVMs are vectors of real-valued numbers whereas are projected to higher dimension space with the help of a function that satisfies the Mercer's condition, the so-called kernel function [115]. Let the training set consists of vectors $x_1, \ldots, x_n$ that belong in the feature space $X \subseteq \mathbb{R}$ $y_i = \pm 1$ are the corresponding labels. Let Φ be the function that projects the datapoints to the higher dimensional feature space H. The kernel function is fundamental component of SVM due to the fact that in order a SVM to be trained one does not need to consider the feature space in its explicit form but needs to consider the inner product between the support vectors and the vectors of the feature space. Therefore, the problem that arises from the high dimensional feature space is alleviated, because it allows the computations to take place in the original feature space of the problem. The use of the kernel functions is usually referred to as the" kernel trick" and it was introduced by [69].

After the datapoints are projected to higher dimension space, SVM tries to find the optimal hyperplane that separates the two classes perfectly. As expected, there can be many more than one separating hyperplane for a specific projection of a dataset Fig. 4.2b . However, the optimal is the one that separates the data with the maximal margin Fig.4.2a. SVMs identify the datapoints near the optimal separating hyperplane which are called support vectors. The distance of the support from the separating hyperplane is called the margin of the SVM classifier.

During the Testing phase, the distance of the unseen samples from the hyperplane is calculated. Depending on a threshold of the value of this distance, a sample is classified to positive class or negative class accordingly. In other words, during this phase only the support vectors are to be identified. More formally, given a set of support vectors $x_1, \ldots, x_l$ and their corresponding labels

$y_1, \ldots, y_l$ the decision function for a novel datapoint x is defined as:

$$f(x,a) = sign(\sum_i y_i {a_i}^0 K(x, x_i) + b \tag{4.7}$$

Where ${a_i}^0$ is a set of parameters need to be optimized in the training set. It should be noted that the non-linear decision function |Equation 4.7 in the input space X is equivalent to the following linear decision function Equation 2.3 in the higher dimensional space H, because of Mercer's condition :

$$f(x,a) = sign(\sum_i \sum_i {}_i y_i {a_i}^0 < \Phi(x_i) * \Phi(x_j) > +b \tag{4.8}$$

The result of the equation is the prediction value of the classifier for the datapoint x and it is the distance of the datapoint from the separating hyperplane. In order one to find the optimal parameters a_i the following function need to be maximized:

$$W(a) = \sum_{i=1}^{i} a_i - \frac{1}{2} \sum_{i,j}^{l} a_j y_i y_j K(x_i, x_j) \tag{4.9}$$

Whereas is subject to constraints:

$$\sum i_{i=1}^{i} a_i y_i = 0 \quad a_i \geq 0, \quad i = 1, 2, \ldots, l$$

Usually, the training data are not perfectly separable even when they are projected in higher dimensional space. Thus, a cost parameter C is used that represents the penalty for datapoints to being on the wrong side of the hyperplane. The value of C is strongly dependent on the dataset so its value can vary significantly, therefore in order to define its optimal value cross-validation is performed on the training dataset. In this case of non-linearly separable data, the constraints for the a_i parameters become:

$$\sum i_{i=1}^{i} a_i y_i = 0 \; a_i \geq C, i = 1, 2, \ldots, l \tag{4.10}$$

The most widely used kernel function are the polynomial kernel:

$$K(x_i, x_j) = (x_i * x_j + 1)^p) \tag{4.11}$$

The radial basis function kernel:

$$(K(x_i, x_j) = e^{(\| x_i * x_j \| 2/2\sigma^2)} \tag{4.12}$$

However, the most widely used kernel function is the linear kernel which is a polynomial kernel of degree 1.Linear SVMs can be optimized to run in significantly smaller times than those using the general polynomial kernel.

Discovering the optimal hyperplane poses a quadratic programming problem which is superliner in the size of dataset. Apart from the kernel trick many methods have been proposed in order the training time to be reduced but the time required still remains a problem.

Apart from improving the training time, one more important consequence of the kernel function is that it defines the shape of the separating hyperplane to be discovered by SVMs. In fact, the hyperplane belongs to the high dimensional feature space H whose geometry is strongly related to the kernel function and for nonlinear choices of the kernel is no longer flat. Therefore, the choice of kernel and its parameters affects the prediction ability of the classifier significantly. It is proven that more powerful the kernel is the better the results are.

In Figure 4.3 we see that the linear SVM cannot find a separating hyperplane whereas the polynomial SVM can since the latter can identify more complicated boundaries. In figure 4.4 where a more complicate dataset is used , we observe that the radial basis function succeeds in separating the dataset whereas the polynomial kernel fails. It worths pointing that very different separating boundaries are discovered by the two kernels. Thus it is important to choose the kernel

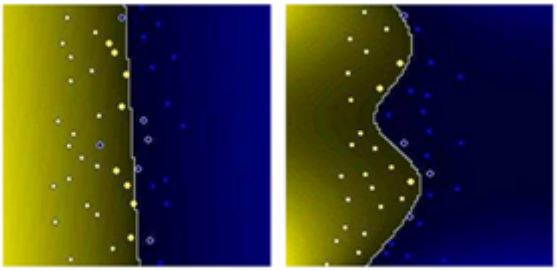


Figure 4.3: Linear SVMs on the left, polynomial SVMs on the right.

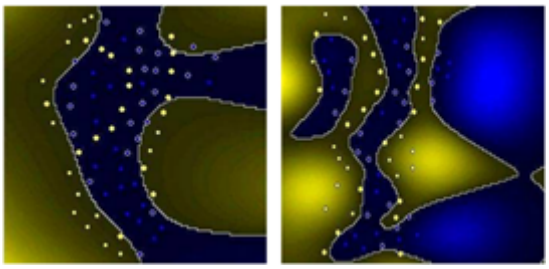


Figure 4.4: Polynomial SVMs on the left, radial basis function SVMs on the right

based of the task at hand. However, the discovery of more complex hyperplanes comes at expense of training time, because more complex kernels results in more computations.

It should also be clarified that the decision values produced using the decision function 4.7 during testing should not be related to probabilities. They can take values in R, unlike the probabilities that lie inside [−1,1]. The absence of probabilistic output from SVMs could be an obstacle in using them in certain applications.

The decision value though is a measure of the confidence an SVM classifier has in its decision for a datapoint. The larger the absolute value, the more confident the classifier is in the decision, because the decision value represents the distance of the datapoint from the optimal hyperplane. However, it must be said that the decision values produced by SVMs with different kernels and/or datasets are not comparable.

Support Vector Machines have significant theoretical advantages over other machine learning methods. They don't require any independence assumptions, unlike Naive Bayes. SVMs are capable of discovering non-linear separating boundaries between classes, while Maximum Entropy can discover only linear ones. Finally, even though SVMs accept only numerical features, categorical features can be used too, by mapping them to numerical features using one out-of-m encoding. In practice, this means that one can use both numerical and categorical features to describe the instances of the problem, which cannot be done easily with Maximum Entropy or Naive Bayes.

4.3.4 Logistic regression

Logistic regression in essence aims to identify the relationship between a binary response and one or more predictor attributes. In logistic regression the response variable is binary, and that distinguishes logistic regression from ordinary linear regression. Unlike the linear model that tries to predict the mean response, logistic regression aims to predict the logit (log-odds) of the response having one particular value versus the other value. If response value only takes values 0 and 1 then the logit of the response with value 1 would be $\frac{(P(Y=1))}{[1-P(Y=1)]}$. Taking the logarithm of this ratio results in a response that varies between $(-\infty, +\infty)$. This in turns results in the following model:

$$logit[P(Y = 1|X)] = \beta_0 + \beta_1 \mathrm{X}_1 + \cdots + \beta_\kappa \mathrm{X}_\kappa \tag{4.13}$$

Where Y is a random variable denoting the response, vector $\mathrm{X}^{\mathrm{T}} = (\mathrm{X}_1, \ldots, \mathrm{X}_\kappa)$denote a collection of k independent predictor variables, and where the β_ι are the parameters to be estimated.

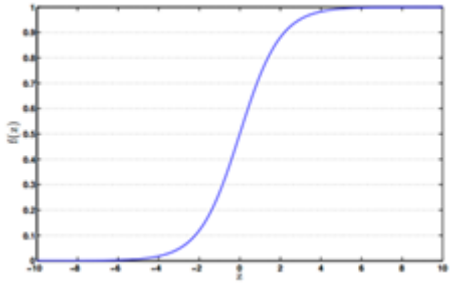


Figure 4.5: The logistic function

The logit transform can be defined as:

$$logit(x) = ln(\frac{x}{1-x}) \tag{4.14}$$

By substituting the equation 4.14 in equation 4.13 the probability $P(Y = 1\big|X)$ will be:

$$P(Y = 1\big|X) = \frac{e^{(\beta_0+\beta_1 X_1+\cdots+\beta_\kappa X_\kappa)}}{1+e^{(\beta_0+\beta_1 X_1+\cdots+\beta_\kappa X_\kappa)}} \tag{4.15}$$

The equation 4.15 is called the logistic regression model, and a maximum likelihood estimation method can be used to identify the parameters in the model. Assume a simplified version of this equation as follows:

$$f(z) = \frac{e^z}{(1+e^z)} \tag{4.16}$$

This logistic function is depicted in the Figure 4.5. By assuming that the value 0 denotes the normal slides and value 1 denotes abnormal slides, the function $f(z)$ represents the estimate for a given value of z.

This idea can be used to utilize a fitted logistic regression model as a classifier. In fact, for any given observation of x_i, in the equation 4.15, the fitted logistic regression will end up with an estimated probability (see equation 4.16) that shows the likelihood of being 1. This estimated probability can be used for binary classification purposes. For this purpose, it is necessary to define a cut − point value and compare the estimated probabilities with this value. Observations having the estimated values of higher than the cut − point, belong to the abnormal class and vice versa. The optimal decisions in the logistic regression are based on the posterior class probabilities $p(y|x)$, and for a binary classification problem such as Pap smear classification these decisions can be written as:y

$$y = \begin{cases} 1, if\ ln = \frac{P(y=1\big|x)}{P(y=0|x} \succ 0 \quad 0, \quad otherwise \end{cases} \tag{4.17}$$

Generally, the probabilities $P(y|x)$ are not known but the possible decisions can be parametrized according to:

$$ln\frac{P(y=1\big|x)}{P(y=0|x)} = f(x,w) = w_0 + X^T w_1 = 0 \tag{4.18}$$

The parameters can be calculated using a maximum likelihood estimation method. Figure 4.6 shows a possible linear decision boundary for classification of a two-class data.

4.3.5 Maximum entropy

Maximum Entropy is considered to be one the most popular models in natural language processing because it deals sufficiently with feature extraction problems that NLP tasks are suffering from. The basic problem is that many methods are producing features that are statistically depended. For instance, the extraction for both unigrams and bigrams leads to dependent features as the occurrence of a words in unigram is by definition part of a bigram. The reason behind the statistical

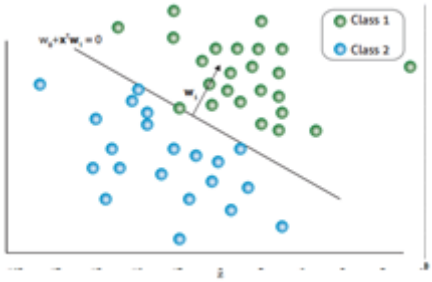


Figure 4.6: Logistic regression linear decision for binary classification

dependency of words is that the words in a text are considered with specific semantic in mind therefore, word-based features in principle will be correlated. It is important that property to be taken into consideration in the formation of a statistical model. As we mention in section 4.3.1 the Naive Bayes makes the assumption that the features are statistical independent. The maximum entropy models on the other hand, are based on the principle that one should use the model which is closet to a uniform distribution unless contrary information shows otherwise. This causes the so-called explaining away effect [126] which means that features which explain the train data best are given high weights.

Formally, the maximum Entropy models aims to maximize the entropy $H(p)$ of the distribution $p(c|x)$:

$$H(p) = -\sum_{c \in C} \sum_{x \in X} p(c\,|\,x)\, logp\,(c\,|\,x) \tag{4.19}$$

The constraints are that $p(c|x)$ needs to be maximal for the correct label and p needs to be a probability distribution. Formulating the Lagrangian and finding an optimum we are getting the model described from the following Equation:

$$p(c\,|\,x) = \frac{exp(\sum_{i \in F} w_f^{(c)} \varphi_i(x))}{\sum_{c \in C} exp((\sum_{i \in F} w_f^{(c)} \varphi_i(x))} \tag{4.20}$$

This exponential model function is called Softmax in the perspective of neural networks as we will see later in this thesis. The feature weights can be calculated by maximizing the model's log-likelihood $LL(w|D)$:

$$LL_w(D) = log \prod_{x,y}$$

For the data on a set of training examples through numerical optimization applying gradient-based methods etc. The log-likelihood function is related to the cross-entropy function which is used for neural network training. The Maximum Entropy model can itself be interpreted as a neural network

4.3.6 K-nearest neighbor

Nearest neighbor algorithms are well known and widely used methods. The nearest neighbor classification is straightforward and intuitive. Given a set of labelled training samples each new unlabeled target sample is assigned to the same label as its nearest neighbor which is defined using a similarity measure. However, the most common nearest neighbor classifier is the so called K-nearest neighbor where k nearest neighbors are used to determine the class of the given target example. In order the classifier to determine the similarity between the given target and its neighbors distance measures are used. The most widely used distances are the absolute distance and the Euclidian distance given from the following equation:

$$d(A, B) = (\sum_{i=1}^{n} [a_i - b_i]^r)^{\frac{1}{r}} \tag{4.22}$$

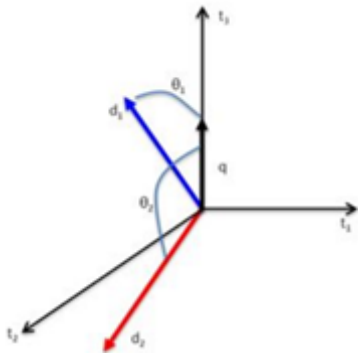


Figure 4.7: Cosine similarity of two document vectors

Where the absolute distance is given when$r = 1$ and Euclidian when $r = 2$ However, in text classification cosine similarity is commonly used in order to measure the similarity of two documents vectors base on the cosine of the angle between two vectors.

As shown in figure 4.7, the similarity between the document d_i and the target document q is given by the cosine of the angle θ_i. Mathematically, the cosine similarity of two document is given as follows:

$$CosSim(d_i, d_k) = \frac{\overrightarrow{d}_j * \overrightarrow{d_k}}{|\overrightarrow{d_j}| * \overrightarrow{d_k}} = \frac{\sum_{i=1}^{t}(w_{ij}w_{ik})}{\sqrt{\sum_{i=1}^{t} wj^2} * \sum_{i=1}^{t} w_{ik}^2} \tag{4.23}$$

Once the k neighbor closest to the target example are found a voting scheme is used to decide the class prediction of the target example. In most implementations of the K-nearest neighbor classifier the majority voting is used. However, in this voting scheme, the k neighbors are considered to be equal indicators of the prediction for the given target. For instance, in a sparse space some neighbors of the target may be far away and to have little or no effect on the target example. In order this problem to be addressed the weighted K Nearest Neighbor was proposed where neighbors close to the target are getting higher weight that the far ones.

The main computation performed by the K-NN algorithm is the sorting of training examples so as the k nearest neighbors of the target example to be founded. This task is computationally expensive so many algorithms have been proposed in order to reduce the computational complexity of this classifier. Usually, the K-NN classifier is used as a baseline due its simplicity and flexibility to be used over different data types.

4.4 Evaluation of machine learning methods

The objective of evaluation of different machine learning methods is to calculate an error for a method in order to compare the methods results with other methods in a given task. The general idea behind the evaluation is to split the data into training, validation, and test set. The method is taught using the training set, validated against a validation set to get the expected error, and finally use test set to see how the method would perform in a real situation. The existence of validation set is crucial due to the fact that comparing classifiers in general is useless because the performance of machine learning algorithms relies heavily on the data used. However, there are different indicators of performance other than error rates due to the fact that each task has its own objective. Therefore, training time, test time, space complexity and computational complexity can be used also as measures.

Cross-validation is used to evaluate how erroneous the methods are in a classification problem. The training data are split into K parts and then again are split into a training set and a validation set that are going to be used in order a method to be trained and validated. Additionally, k fold cross validation splits the training data into K equally sized sets. One set is used as a validation set, while others are used as a training set (size K-1). This is then repeated so that each split

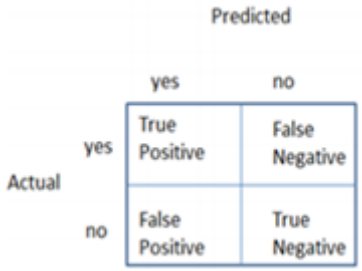


Figure 4.8: The confusion matrix

is being a validation set once, thus giving K amount of repetitions. The errors are gathered and averaged. The errors in cross-validation techniques are slightly dependent as the data will overlap with itself.

In classification the most known tool for measuring the classification performance is the confusion matrix as shown in the figure. Basically, the confusion matrix is a table that shows the numbers of the correctly and incorrectly labeled examples. The size of the matrix is the number of classes times the number of classes. However, the practical case in to compare one class over all others so the size will be two times two.

A confusion matrix consists for the following parts:

- True Positive (TP): correctly predicted as a positive example
- True Negative (TN): correctly predicted as a negative example
- False positive (FP): incorrectly predicted as a positive example
- False negative (FN): incorrectly predicted as a negative example

Moreover, from the confusion matrix are derived many metrics that used to evaluate the performance of a classifier. The most common of all is the accuracy. Accuracy indicates the proportion of the correctly predicted samples. So, the accuracy is given as follows:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.24}$$

In most applications, only the accuracy is the strongest indicators of the performance and the only used metric as in this thesis. However, there tasks that accuracy fails to indicate the performance of the classifier thus more performance measurements are needed. In sentiment analysis, precision and recall are broadly used. The precision means, how well the model is to be trusted. The precision tells the proportion of the positive examples that are truly positive.

The precision is given as follows:

$$precision = \frac{(TP)}{(TP + FP)} \tag{4.25}$$

On the other hand, the recall indicates how large the proportion of the correctly labeled examples is from the total positive examples:

$$recall = \frac{TP}{TP + FN} \tag{4.26}$$

Another, metric used in this thesis is the F measure that is a harmonic mean of the recall and the precision and it defined as:

$$F - measure = \frac{2 * precision * recall}{precision + recall} \tag{4.27}$$

A common visualization tool for the performance evaluation is the receiver operating characteristic curve, or the ROC curve. The ROC curve indicates the tradeoff between the detection. Most

classifiers fall between the perfect classifier and the classifier with no predictive value. Usually, a measure called area under the ROC, or AUC, is calculated to see how close a classifier is to a perfect classifier. of true positives and the false positives.

Chapter 5

Sentiment Analysis

In this chapter, we will introduce the general principles and methods of Sentiment Analysis and familiarize the reader with concepts used later in this Thesis. Before we discuss sentiment analysis in depth, is crucial to present some definitions of opinions. Liu [68] defines opinions as "subjective expressions that describe people's sentiment, appraisals or feelings towards entities, events, and their properties". Moreover, he is proposing a discrimination between a direct opinion, that gives a positive or negative opinion about an entity and a comparative opinion, in which a positive or negative opinion is implied by comparing two entities.

Another definition of opinions came from Kim and Hovy where the opinion is defined as a combination of four factors: topic, holder, claim and sentiment where Liu defines five factor of a direct opinion: object, features, orientation, holder and time. In case of Kim and Hovy [56],the opinion holder expresses a claim about a certain topic and associates this claim with a certain sentiment. On the other hand, in case of Liu , the opinion holder distinguishes some features of a certain entity and associates a certain sentiment orientation to them in a specific time.

5.1 Definition

According to Merriam-Webster dictionary [76] the word sentiment has three layers of meanings:

- Predilection or opinion.
- Emotion or refined feeling.
- Idea colored by emotion.

A sentiment can be described as an opinion or attitude stated by an individual, the so-called opinion holder for an entity, the target. In [96] the authors distinguishes attitudes from emotions. Attitudes are defined as ""relatively enduring, affectively colored beliefs, preferences towards objects or persons" while emotions are "brief episodes of synchronized responses" as a reaction to an external event [96]. Therefore, what distinguishes sentiment analysis from emotion analysis, is that in emotion analysis the internal state of the writer is of interest and not his/her claim for a specific target. The degree and the direction of the expressed sentiment are called polarity.

> Sentiment analysis is the field of study that analyses peoples opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organisations, individuals, issues, events, topics, and their attributes[68]

Sentiment analysis in the scope of computational linguistics, is tackled as a classification problem; text is classified into two classes; positive and negative. But there are plenty of research done in which the classification process appears as multi-categorization task instead of a binary where a multi points scale is proposed to rank text [88]. Sentiment classification key components can be summarized to the following categories: tasks, features, techniques and applications [1]. There are four main tasks that are tackled in present day sentiment analysis research: subjectivity analysis, sentiment classification, opinion summarisation, and opinion extraction and mining. Sentiment

analysis can be carried out at a number of levels: words, phrases, sentences and (sets of) documents. The features can be syntactic, semantic, link based and stylistic. Techniques can be either machine learning based, link analysis or similarity scores of bases.

Research concerning the emotional orientation of linguistic expressions has a long history in linguistics. An important related concept that has been introduced early is connotation, the additional emotional or cultural meaning of an expression . The connotation is a strong indicator of expressing sentiment. However, the computational approaches to sentiment appeared in the early 90s. The term Sentiment Analysis appeared for the first time in [27] there was some earlier work on the interpretation of metaphors, sentiment adjectives, subjectivity, viewpoints, and affects [26, 88]. The breakthrough of statistical models for large-scale sentiment analysis was introduced by [89] who showed that user-supplied polarity ratings of product reviews can be predicted automatically with high accuracy through supervised classification. Since then, numerous sentiment analysis approaches on a broad range of linguistic levels have been proposed. These range from low-level units like words and phrases [120] to sentences [87] to fine-grained document analysis [50]. Moreover, apart from statistic methods, sentiment analysis is supported by vocabulary lexical resources such as polarity lexicons [131].

An important subject that is examined in sentiment analysis is subjectivity analysis, where the objective is to recognize if a statement is reflecting an individual's private state or it express objective judgment.The main difficulty of classifying text as positive or negative depends whether it expresses a fact or an opinion. The classification of opinionated text is much harder than factual one because opinionate text usually is expressed in a more informal way such as sarcasm, subjective language, emoticons and so on. In [68] the author claims that much of research in textual sentiment analysis was conducted on factual text because in the past researchers, were not able to obtain huge amount of user-generated content as we can now. Now, is almost impossible for companies or humans to retrieve and summarise the state of general opinion due to the tremendous size of media data and their diversity. Sentiment analysis emerged from this need. Sentiment analysis studies till today have been applied in numerous different types of texts and domains. The choice of domain, is based on the practical applicability of sentiment analysis. For instance, customer reviews are of interest of companies who would like to have feedback about what the customers beliefs about their product or their marketing strategy. Social media (blogs, internet forums, social networking websites and others) are a hot topic in many marketing and media studies as they are not only a valuable source of opinion-related information but also a medium where opinions are forged.

5.2 Concepts in Sentiment Analysis

Different authors have dealt with the problem of sentiment classification in different ways. Sentiment classification can be formulated either as two separate classification problems or as a three-class classification problem [68]. When formulated as a two separate classification problems, the first problem is to determine if a piece of text (e.g. a document) is subjective or objective, that is, if it expresses an opinion or not. This type of problem is called subjectivity classification [43, 129, 131]. The second classification problem is to classify the subjective sentences into positive or negative. This binary classification task of labelling a document as expressing either an overall positive or an overall negative opinion is called (sentiment) polarity classification [88] .

5.2.1 Polarity

The sentiment determination is probably the most studied in sentiment analysis literature. It consists of identifying polarity or orientation. In its simplest form, it corresponds to the binary orientation (positive or negative) of a subjective statement without taking into consideration the external context of a sentence or document. Several types of scales have been used in sentiment analysis research, going from continuous scales [4] to discrete ones [116]. Generally, it is not proven how many points are needed on the scale, but the scale has to ensure that a fine-grained categorization of subjective words is with respect to human judgments.

Usually, polarity is mapped to the [-1,1] interval where -1 denotes the most negative polarity possible and 1 the most positive. However, there is some ambiguity regarding the interpretation

of the center of the scale. It can be seen as neutral polarity or denotes a more balanced mix of positive and negative content. It has been proven that even for humans are difficult to assess whereas a statement is neutral or not, which is why in most cases this category is omitted in order to simplify the problem.

However, in 2011 Socher [110] introduced a more complex polarity problem whereas he proposed the use of the scheme from the experience project which collects user-submitted stories. In this case, users can vote for one of six tags:you rock, tehee, I understand, sorry, hugs, and wow, just wow, which scientifically bordering emotion analysis spectrum. In this thesis, we are concerned with binary polarity prediction problems, thus focusing only on the two extreme points (positive and negative) of the polarity scale.

Sentiment bearing words may be ambiguous, leading to different polarities in different contexts. Therefore, exists two main trends in polarity: the prior polarity when the determination of the polarity value of a word does not take into account the context and the contextual polarity where the context is taken into account [131]. Contextual influences of a words polarity may arise for many reasons. First, the perception of the meaning of a word could be different across domains. For instance, the word scary can express a positive sentiment when is used for a movie but negative when is used for a hotel review. Second, in most of the cases same word might have different meanings across cultures thus the world knowledge of a sentiment analysis system is of great importance.

5.2.2 Subjectivity

In terms of philosophy, it has been pointed out that there is a clear distinction between subjective impressions of individual and objective reality. One can notice this distinction manifest itself in human communication leading to subjective or objective language. Whether a statement is subjective or objective depends on their verifiability. On the other hand, the term subjectivity in computation linguistics comes with different senses and mainly is a strong indicator for both sentiment and polarity. Usually in computational linguistic the subjectivity is strongly related to the point of view. In our presentation of subjectivity, we will focus on the distinction between objective and subjective statements. Wiebe and colleagues have devoted considerable effort to finding indicators of subjectivity in sentences [128, 129, 131]. In order to identify the subjectivity, they proposed a set of subjectivity indicators either lexical or syntactic. The set of lexical indicators consists of psychological verbs or verbs of judgment and adjectives that have been previously annotated for polarity. The syntactic clues are learned from manually annotated data [128, 129]

Liu points out that a subjective sentence may not express any sentiment and in objective sentences opinions or sentiment can be implied due to desirable and undesirable facts [135]. Additionally, subjective or objective texts are hardly ever stated explicitly thus this complicates the automatic processing of those texts. Another challenging aspect of subjectivity analysis is, that not always the entire texts are either objective or subjective. Even a single sentence may contain factual information and some subjective evaluation of it.

However, a number of studies demonstrate reasonable success in subjectivity analysis. The subjectivity task has been used as a criterion to denote the importance of an expression regarding the sentiment analysis. Pang and Lee [87]for example propose subjectivity-based selection for summarizing reviews. Subjectivity analysis is used by many researchers as a pre-processing step to filter the data for sentiment classification. [45, 67]

All in all, the definition of subjectivity given above is not universally accepted. While in this thesis, the classical definition is considered many related concepts are presented under the umbrella of subjectivity. For instance, the approach by Pang and Lee [87] uses proxy data in place of any formal definition of subjectivity, showing that practical concerns are often of higher importance than formal correctness.

5.3 Types of sentiment analysis

Sentiment analysis is concerned with predicting the sentiment orientation of unseen data. Since, this term covers a broad range of problems thus many approaches have been proposed. In the

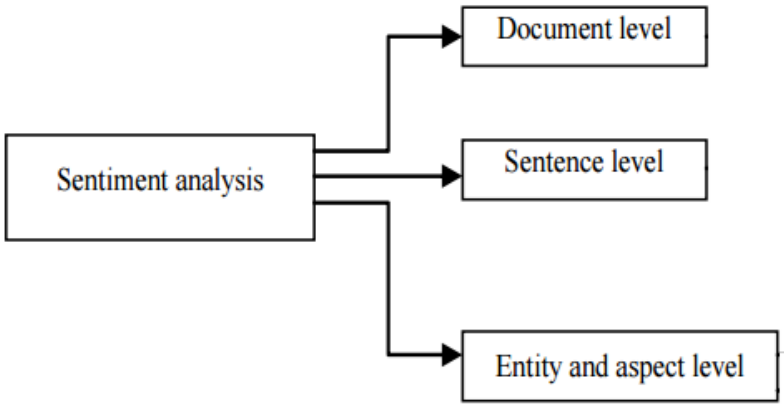


Figure 5.1: Types of Sentiment Analysis

following paragraphs, we are going to discuss three general types of sentiment analysis problems. There is a highly diverse set of methods tackling these problems which cannot easily be summarized briefly. In general, they can be divided into two major classes: rule-based and statistical approaches. In this thesis, we focus mainly on statistical approaches.

In today's research many different views on sentiment analysis exist, leading to different tasks. The most widely used method to categorize those methods is the granularity of the analysis. Sentiment analysis can be performed on multiple linguistic levels figure.5.1

- At the document level, the task is to classify if a whole opinionates document conveys positive or negative sentiment
- At the sentence level, the task is to classify if an individual sentence conveys positive or negative sentiment.
- At the aspect level (the entity level), the task is to classify the sentiment of individual sentences or phrases intended towards certain entities or aspects

In this section, we will introduce these levels based on the overview by [68].

5.3.1 Document level

Document level sentiment analysis aims to predict the overall polarity expressed in a document. Typically, the documents on which this type of analysis is performed are those in which authors criticize a single entity, such as review of products, hotels and movies. Usually, in addition to the review text users also supplies a rating on a predefined polarity scale. It is worth pointing that in the review domain movies reviews is much longer than others thus the average document length may vary between domains.

The task of prediction document-level sentiment polarity can be performed as standard text classification problem. Therefore, well-established techniques can be used, such as maximum entropy classification [89]. Two major assumptions involved in the text classification approach. To begin with, it is assumed that the whole text contains a single target, the product that is the subject of the review. The authors are assumed to be the opinion holders. However, these assumptions might not hold in all domains. For instance, in the news domain opinions by different holders about many targets can be presented by the news reporters. However, in movies reviews those assumptions hold.

Additionally, documents are long enough thus a violation of the aforementioned assumptions maybe disregarded as noise. Therefore, the document classification problem relies on the sufficient availability of clear indicators of sentiment.

A commonly used strategy in document-level sentiment analysis is to use rating provided by users for supervision. As we mentioned above, many ratings schemes exist so is essential this rating

to be translated in a common simplified scale. The use of users rating as labels, is an appealing idea since its eliminates manual annotation but is far from perfect. It has been shown that when used manually annotated reviews better recommendations are produced.

In literature, many causes for this effect has been pointed out. First, users don't follow any standards concerning the rating procedure. Moreover, many online reviewing schemes correlate star rating with verbal assessments but usually, there are no detailed guidelines that give specific instructions for the rating process. Second, some readers see the review text as complementary to the rating, and thus, the rating may be inconsistent with the views expressed in the text. These results and observations show that although sentiment classification is often described as supervised, caution must be taken with viewing star ratings as a gold standard. Overall, the document-level approach has been found to be too coarse for many practical applications. Thus, finer-grained analysis levels have been proposed, which we review next.

5.3.2 Sentence/Clause Level

Documents usually convey a clear overall polarity. However, smaller linguistic units such as sentences or clauses are significantly harder to be analyzed. Again, the problem of polarity orientation of a sentence can be treated as a text classification task where the text is represented as word-based features and then those features are fed to a statistical model. To this point, the most prominent sentence level approach is using a combination of naïve Bayes and support vector machine [125]. Sentence level sentiment analysis relies heavily on sentiment polarity bearing words. Therefore, taking into consideration that a sentence is scientifically shorter than documents it should be expected that not every sentence contains actually a sentiment bearing word. Moreover, sentences may also be ambiguous when read out of context, or may contain expressions of mixed sentiment. An early formal model for coping with complex polarity structure was introduced by Polanyi and Zaenen [94]. In their model, polarity-bearing words are modified by the so-called polarity shifters.

In recent literature, has been shown that sentence sentiment analysis, follows a hierarchical compositional structure formalized through constituency or dependency trees. Each node of the tree is representing the polarity; the overall sentence polarity is given at the root node. On-terminal polarities are the result of shifter like operations recursively carried out at each node.

5.3.3 Entity/Aspect Level

Liu [68]defines an entity as any object about which sentiment is expressed. An entity constitutes a type of target which was introduced previously. An aspect is any property or part of the target. We discuss the notions of entities and aspects briefly as they are beyond of the scope of this thesis. The fundamental difference in this approach is that it does not focus on linguistic units for analysis. It mainly focuses on collecting information concerning a specific entity in order a prediction to be made. This task has been introduced as fine-grained sentiment analysis [132] because relations between sentiments, targets or opinion holders have to be extracted. Therefore, this type of analysis is deeper than the previous methods because knowledge about the target is to be found in order ambiguities to be resolved and not linguistic units. Therefore, entity-level sentiment analysis requires significantly more complicated machine learning tasks.

5.4 Sentiment analysis approaches overview

In literature exists three different methods that tackle the problem of sentiment analysis: machine learning methods,lexical based methods and linguistic analysis [119].

Lexicon based approach is based on the construction of a lexicon which is a structure that keeps track of words and possible information about them. The words are usually referred as lexical items [13]. After the construction of the lexicon, the overall polarity of the text is then calculated by a possible weighted count from lexical items exist in text [55]. Dictionaries for lexicon-based approaches can be created manually or automatically using seed words in order to expand the list of words [43, 120, 121] .The main disadvantage of this approach is that strongly depends on the goodness of the lexical resource that relies on.

In linguistic approach, syntactic characteristics of words, phrases, negations and the structure of the text are considered to estimate the text orientation. In literature, this approach usually incorporates and a lexicon-based method as can been seen in [118–120]. The most known linguistic approach is the parts of speech tagging (PoS). With PoS syntactic patterns or categories of words are defined [13]. Numerous patterns can be extracted and be used as phrases. Usually those patterns are indicators of either a sentiment or a topic. Those patterns are represented as n-grams. N-gram can be considered as a sequence of n words from a given sequence of speech.

Last but not least, are the machine learning techniques. The machine learning techniques are using the principles and classifiers we mentioned in the previous chapters. Although, the majority of research is conducted using Machine learning methods this does not mean that the machine learning has no weaknesses. The several algorithms vary in their ability to generalize over large set of patterns, and sometimes occur patterns that the algorithm has not seen before and is more likely to be ignored. But in language, frequently occurring patterns are rare, and rarely occurring patterns are predominant.

The aforementioned approaches can be used separately or combined. Machine learning approaches can incorporate linguistics approaches so that the selected features for machine learning algorithm can be only features of one Pos kind. In many studies have been shown that adjectives usually are strong indicators of sentiment and is not rare lexicon to be built from adjectives. Of course, this does not imply to ignore the importance of other parts of speech (verbs or nouns)

In this thesis, machine learning classification methods are considered. Since these methods accomplished state-of -the arts results in the movie reviews domain. The approaches described above such as linguistic Lexicon base approach is beyond the scope of this thesis.

5.4.1 Sentiment Analysis using machine learning techniques

With the increasing need of information organization and knowledge discovery from text data, many supervised learning algorithms have been used for text document classification. Among these methods, naive Bayes and Support Vector Machines (SVM) are always in the comparison list. Naive Bayes – a generative classifier – is considered a simple but effective classification algorithm [79]. SVM – a discriminative classifier – is considered the best text classification method to date. The key point of using machine learning for sentiment analysis lies in engineering a representative set of features (see Chapter 2). The first, pioneering in this direction, was [89] were conducted polarity classification of reviews using feature engineering methods. The techniques which they explored are support vector machines(SVMs), naïve Bayes (NB) and maximum entropy (ME) classifiers. They experiment by using different feature sets such as unigrams, bigrams, binary and term frequency feature weights and others. The outcome of their observation was that sentiment classification is not that easy than standard topic-based classification they also concluded that using an SVM classifier with binary unigram-based features produces the best results. In their later work, they introduced a twofold sentiment classification process; First, they detect and remove objective parts of the document and then they apply a polarity classifier on the remain parts [87]. This exploited text coherence with adjacent text spans which were assumed to belong to the same subjectivity or objectivity class. They used graph representation of documents where the sentences were the nodes and association scores between them as edges. Two additional nodes represented the subjective and objective poles. The weights between the nodes were calculated using three different, heuristic decaying functions. Finding a partition that minimized a cost function separated the objective from the subjective sentences. They reported a statistically significant improvement over a NB baseline using the whole text but only slight increase compared to using a SVM classifier on the entire document [88].

Turney[120]introduced an unsupervised learning algorithm for classifying a review as “recommended” (thump up) or “not recommended” (thump down). First, they extracted phrases containing adjectives or adverbs. Secondly, they calculated the semantic orientation using Pointwise Mutual Information (PMI). Finally, they classified the reviews based on the average semantic orientation of the phrase. Turney and Littman [121] also introduced a method for inferring the semantic orientation from associations. The relation between a word and a set of positive or negative words was measured using two different statistical measures: PMI and Latent Semantic Analysis (LSA).

Mullen and Collier [83] worked on the same dataset used by [89]. They calculated the average

rating for the whole collection. Then, the reviews under the average rating were classified as negatives and those above the average rating were classified as positives. They investigated various features including Combination of Turney value, the three text-wide Osgood values, word unigrams or lemmatized unigrams. In addition, they accomplished experiments over a movie reviews corpus downloaded from the Pitchfork Media. In this case, they extracted the same features and extra features based on the movie domain. The machine learning algorithm used was the SVM. They concluded that the combination of unigrams and lemmatized unigrams outperforms the models which do not use this kind of information.

Prabowo and Thelwall [95] applied SVM with combined methods to classify reviews from different corpora. One of these datasets was downloaded from Pang and Lee (2004) and it includes 1,000 positive and 1,000 negative samples. Several classifiers were used: General Inquirer Based Classifier (GIBC), Rule-Based Classi- fier (RBC), Statistics Based Classifier (SBC) and SVM. They accomplished a hybrid classification, where if one classifier fails to classify a document, the classifier passes the document onto the next classifier until the document is correctly classified or no other classifier remains. The results indicated that SBC and SVM improve their effectiveness in the hybrid classification.

Zhang [136] classified sentiment using machine learning (NB and SVM) for restaurant reviews written in Cantonese. They studied the effects of feature representations and feature size on the classification performance. Experiments were performed on 1500 +ve and 1500 -ve reviews. They experimented with different feature representations like unigram, unigram_freq, bigram, bigram_freq, trigram, and trigram_freq and varying number of features in the range of 50 to 1,600 features.They find that accuracy is influenced by interaction between the classification models and the feature options. The naive Bayes classifier achieves as well as or better accuracy than SVM. Character-based bigrams are proved better features than unigrams and trigrams in capturing Cantonese sentiment orientation. The highest accuracy reported was 95.67% using NB for 900–1100 features. Future work can be integration of automatic review mining technologies to search engines. Read [98] studied the dependence of naive Bayes and SVM classification models on domain and time. SVM did not beat naive Bayes in sentiment classification as in topic classification. However, Engstrom [29] showed that the bag-of-features approach is topic-dependent. A classifier trained on movie reviews is unlikely to perform as well on (for example) reviews of automobiles. Turney [120] noted that the unigram unpredictable might have a positive sentiment in a movie review (e.g. unpredictable plot), but could be negative in the review of an automobile (e.g. unpredictable steering).

Saleh [104] carried out twenty seven sentiment classification experiments using SVM with various feature selection methods. Experiments were performed on three well benchmarked datasets. Using 10-FCV, the best classification accuracy reported were 85.35%, 73.25%, and 91.51% for pang corpus using binary occurrences and trigrams, Taboada corpus using term frequency-inverse document frequency (TF-IDF) and trigrams, SINAI corpus using TFIDF and bigrams respectively. Further experiments can be performed to observe the results affected by rating reviews

Moraes [81] compared popular machine learning approaches (SVM and NB) with an ANN-based method in the context of document-level sentiment classification [68]. In comparison with the sentiment classification literature, the main contributions of thid work are: (i) a comparison of a dominant and a computationally efficient approach (SVM and NB, respectively) with an ANN-based approach under the same context; (ii) a comparison involving realistic contexts in which the ratio of positive and negative reviews is unbalanced.Four datasets were chosen for this purpose benchmark movies review dataset [87] and reviews on three distinct product domains: GPS, Books, and Cameras. For unbalanced dataset, performances of both classifiers ANN and SVM were affected. Information gain as feature selection method did not help yield good accuracy for more than 1,000 features. Therefore, these classifiers should be tested on given dataset using different feature selection methods. Wang [124]compared the performance of three popular ensemble methods viz. bagging, boosting, and random subspace based on five base learners namely NB, ME, Decision Tree, KNN, and SVM for sentiment classification. They experimented with ten different datasets and reported better accuracy over base learners at the cost of computational time. Ensemble method can be further validated on large dataset, and feature construction based on linguistic method can also be considered

Dave [27] describes a tool for sifting through and synthesizing product reviews, automating the

sort of work done by aggregation sites or clipping services. They begin by using structured reviews for testing and training, identifying appropriate features and scoring methods from information retrieval for determining whether reviews are positive or negative. These results perform as well as traditional machine learning methods. Then they used the classifier to identify and classify review sentences from the web, where classification is more difficult. However, a simple technique for identifying the relevant attributes of a product produces a subjectively useful summary. Unlike Pang's research, they obtained the best accuracy rate with word bigram-based classifier on their dataset. This result indicates that the unigram-based model does not always perform the best and that the best settings of the classifier is dependent on the data.[25]classified sentiments using SVM by using different feature selection methods. Experiments were performed on two corpora on digital camera and multi-domain dataset. SVM was trained on three collections of features set based on domain free, domain dependent, and sentiment features. Information Gain (IG) was applied to reduce the number of features for different combination of features. The reduced features set performed better on multi-domain dataset than digital camera dataset, and yielded an accuracy of 84.15% for kitchen appliance. The proposed feature selection methods should be tested on bigger dataset and compared with other statistical based feature selection method

In [125], authors showed that the inclusion of word bigram features gives consistent gains on sentiment analysis tasks,for short snippet sentiment tasks, NB actually does better than SVMs (while for longer documents the opposite result holds). Based on these observations, they identified simple NB and SVM variants which outperform most published results on sentiment analysis datasets, sometimes providing a new state-of-the-art performance level [70] 91.4%. The authors showed the usefulness of bigram features in bag of features sentiment classification has been underappreciated, perhaps because their usefulness is more of a mixed bag for topical text classification tasks. Then they performed a distinction between short snippet sentiment tasks and longer reviews, showing that for the former, NB outperforms SVMs. Contrary to claims in the literature, they proved that bag of features models are still strong performers on snippet sentiment classification tasks, with NB models generally outperforming the sophisticated, structure-sensitive models explored in recent work. Furthermore, by combining generative and discriminative classifiers, they presented a simple model variant where an SVM is built over NB log-count ratios as feature values, and show that it is a strong and robust performer over all the presented tasks. Finally, they confirmed the wellknown result that MNB is normally better and more stable than multivariate Bernoulli NB, and the increasingly known result that binarized MNB is better than standard MNB. In line with Wang's and Manning's work is [77]where they achieved new state-of-the art results on the Stanford Large movie dataset [70]. Authors compared several different approaches and proved that model combination performs better than any individual technique. The ensemble best benefits from models that are complementary, thus having diverse set of techniques is desirable. The vast majority of models proposed in the literature are discriminative in nature, as their parameters are tuned for the classification task directly. In this work, they boosted the performance of the ensemble by considering a generative language model. To this end, two language models were trained, one on the positive reviews and one on the negative ones, and use the likelihood ratio of these two models evaluated on the test data as an additional feature. Authors assumed that a positive review will have higher likelihood to be generated by a model that was trained on a large set of positive reviews, and lower likelihood given the negative model.

Xia et al. [54] ensembled feature sets and machine learning for sentiment classification. Two types of feature sets namely "POS based features" and "the world-relation based feature sets" have been designed. These feature selection methods were ensembled with NB, ME, and SVM using 3 techniques viz. fixed combination, weighted combination and meta-classifier combination. Word relation based weighted classifier yielded accuracy of 87.7% and average 85.15% on Pang and Lee [89]dataset.

In [75], authors investigated a structured model for jointly classifying the sentiment of text at varying levels of granularity. Inference in the model is based on standard sequence classification techniques using constrained Viterbi to ensure consistent solutions. The primary advantage of such a model is that it allows classification decisions from one level in the text to influence decisions at another. Experiments show that this method can significantly reduce classification error relative to models trained in isolation. Structured models have previously been used for sentiment analysis. Choi [14] use CRFs to learn a global sequence model to classify and assign sources to opinions.

Mao [73]) used a sequential CRF regression model to measure polarity on the sentence level in order to determine the sentiment flow of authors in reviews.

5.4.2 Sentiment analysis using neural networks

Feature engineering is important but labor intensive. It is therefore desirable to discover explanatory factors from the data and make the learning algorithms less dependent on extensive feature engineering. With the rapid growing of deep learning (representation learning [46]), many recent studies focus on learning the low-dimensional, dense, and real-valued vector as text features for sentiment analysis without any feature engineering. Existing deep learning methods for sentiment classification typically include two stages. In the first stage, they learn word embeddings from text corpus. In the second stage, word embeddings are applied to producing the representations of sentences/documents with semantic composition. Existing composition learning approaches are typically based on the principle of compositionality,which states that the meaning of a longer expression (e.g., a sentence or a document) comes from the meanings of its constituents and the rules used to combine them.

Neural embeddings were first proposed by [7], in the form of a feed-forward neural network language model. Modern methods use a simpler and more efficient neural architecture to learn word vectors (word2vec: [78], GloVe: [93]), based on objective functions that are designed specifically to produce high-quality vectors. Neural embeddings learnt by these methods have been applied in a myriad of NLP applications, including initialising neural network models for objective visual recognition as well as directly modelling word-to-word relationships [78, 105, 137].

Word2vec was proposed as an efficient neural approach to learning high-quality embeddings for words. Negative sampling was subsequently introduced as an alternative to the more complex hierarchical softmax step at the output layer, with the authors finding that not only is it more efficient, but actually produces better word vectors on average. The objective function of word2vec is to maximise the log probability of context word (w_O) given its input word (w_I), i.e. $log\ P(w_O|w_I)$. With negative sampling, the objective is to maximise the dot product of the w_I and w_O while minimising the dot product of wI and randomly sampled "negative" words. There are two approaches within word2vec: skip-gram ("sg") and cbow. In skip-gram, the input is a word and the output is a context word. For each input word, the number of left or right context words to predict is defined by the window size hyperparameter. cbow is different to skip-gram in one aspect: the input consists of multiple words that are combined via vector addition to predict the context word

Paragraph vectors, or doc2vec, were proposed by [63] as a simple extension to word2vec to extend the learning of embeddings from words to word sequences. doc2vec is agnostic to the granularity of the word sequence it can equally be a word n-gram, sentence, paragraph or document. In this thesis, we use the term "document embedding" to refer to the embedding of a word sequence, irrespective of its granularity. There are two approaches within doc2vec: dbow and dmpv.

Dbow works in the same way as skip-gram, except that the input is replaced by a special token representing the document (i.e. vwI is a vector representing the document). In this architecture, the order of words in the document is ignored; hence the name distributed bag of words. dmpv works in a similar way to cbow. For the input, dmpv introduces an additional document token in addition to multiple target words. Unlike cbow, however, these vectors are not summed but concatenated (i.e. v_{w_I} is a concatenated vector containing the document token and several target words). The objective is again to predict a context word given the concatenated document and word vectors. Although Mikolov found that as a standalone method dmpv is a better model, others have reported contradictory results. doc2vec has also been reported to produce sub-par performance compared to vector averaging methods based on informal experiments. Additionally, authors reported state-of-theart results over a sentiment analysis task using doc2vec, others (including the second author of the original paper in follow-up work) have struggled to replicate this result.

Socher [100] present Sentiment Treebank. It includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences and presents new challenges for sentiment compositionality. In order to adress those challenges , authors proposedthe Recursive Neural Tensor Network. Trained on on the new treebank ,this model outperforms all previous methods on several metrics. It pushes the state of the art in single sentence positive/negative classification from

80% up to 85.4%. The accuracy of predicting fine-grained sentiment labels for all phrases reaches 80.7%, an improvement of 9.7% over bag of features baselines. Lastly, it is the only model that can accurately capture the effects of negation and its scope at various tree levels for both positive and negative phrases. However, the RecursiveNN captures the semantics of a sentence via a tree structure. Its performance heavily depends on the performance of the textual tree construction. Moreover, constructing such a textual tree exhibits a time complexity of at least $O(n)^2$, where n is the length of the text. This would be too time-consuming when the model meets a long sentence or a document. Furthermore, the relationship between two sentences can hardly be represented by a tree structure. Therefore, RecursiveNN is unsuitable for modeling long sentences or documents.

More recently, [59] proposed skip-thought as a means of learning document embeddings. Skip-thought vectors are inspired by abstracting the distributional hypothesis from the word level to the sentence level. Using an encoder-decoder neural network architecture, the encoder learns a dense vector presentation of a sentence, and the decoder takes this encoding and decodes it by predicting words of its next (or previous) sentence. Both the encoder and decoder use a gated recurrent neural network language model. Evaluating over a range of 8 tasks,semantic relatedness, paraphrase detection, image-sentence ranking, question-type classification and 4 benchmark sentiment and subjectivity datasets, the authors found that skip-thought vectors perform very well against state-of-the-art task-optimized methods. [130] proposed a more direct way of learning document embedding's, based on a large-scale training set of paraphrase pairs from the Paraphrase Database (PPDB:[108]). Given a paraphrase pair, word embeddings and a method to compose the word embeddings for a sentence embedding, the objective function of the neural network model is to optimize the word embeddings such that the cosine similarity of the sentence embeddings for the pair is maximized. The authors explore several methods of combining word embeddings, and found that simple averaging produces the best performance.In fact , they compared six compositional architectures, evaluating them on annotated textual similarity datasets drawn both from the same distribution as the training data and from a wide range of other domains.They found that the most complex architectures, such as long short-term memory (LSTM) recurrent neural networks, perform best on the in-domain data. However, in out-of-domain scenarios, simple architectures such as word averaging vastly outperform LSTMs. Addionionally, in order to understan how these architecures are compared they used them on three supervised NLP tasks namely : sentence similarity, entailment, and sentiment classification. They proven that averaging models perform well for sentence similarity and entailment, outperforming LSTMs. However , in terms of sentimetnt classification on the Standord Sentiment Tree bank the LSTM exceed the previous state-of-the art results.

In [117] authors indrodused a tree-LSTM , a generalization of LSTMs to tree-structured network topologies.Tree-LSTM, composes its state from an input vector and the hidden states of arbitrarily many child units. The standard LSTM can then be considered a special case of the Tree-LSTM where each internal node has exactly one child They proved that Tree-LSTM outperform all existing systems and strong LSTM baselines on two tasks : predicting the semantic relatedness of two sentences (SemEval 2014, Task 1) and sentiment classification Stanford Sentiment Treebank). A variation of LSTM is proposed in [86] where the authors used a recurrent neural networks (RNN) with Long Short-Term Memory (LSTM) cells. The proposed LSTM-RNN model sequentially takes each word in a sentence, extracts its information, and embeds it into a semantic vector. The LSTM-RNN is trained in a weakly supervised manner on user click-through data logged by a commercial web search engine. Visualization and analysis are performed to understand how the embedding process works. The model is found to automatically attenuate the unimportant words and detect the salient keywords in the sentence. Furthermore, these detected keywords are found to automatically activate different cells of the LSTM-RNN, where words belonging to a similar topic activate the same cell. As a semantic representation of the sentence, the embedding vector can be used in many different applications.A comparison with a well known general sentence embedding method, the Paragraph Vector, performed. The results show that the proposed method significantly outperforms Paragraph Vector method for web document retrieval task.Similar to the recurrent models mentioned in [86], The DSSM [51] and CLSM [108] models, developed for information retrieval, can also be interpreted as sentence embedding methods. However, DSSM treats the input sentence as a bag-of-words and does not model word dependencies explicitly. CLSM treats a sentence as a bag of n-grams, where n is defined by a window, and can capture local word

dependencies. Then a Max-pooling layer is used to form a global feature vector. Methods in [21] are also convolutional based networks for Natural Language Processing (NLP). These models, by design, cannot capture long distance dependencies, i.e., dependencies among words belonging to non-overlapping ngrams.

Authors in [24] presented two approaches to use unlabeled data to improve Sequence Learning with recurrent networks. The first approach is to predict what comes next in a sequence, which is a language model in NLP. The second approach is to use a sequence autoencoder, which reads the input sequence into a vector and predicts the input sequence again. These two algorithms can be used as a "pretraining" algorithm for a later supervised sequence learning algorithm. In other words, the parameters obtained from the pretraining step can then be used as a starting point for other supervised training models. In their experiments, found that long short term memory recurrent networks after pretrained with the two approaches become more stable to train and generalize better. With pretraining, they were able to achieve strong performance in many classification tasks, such as text classification with IMDB, DBpedia or image recognition in CIFAR-10.

Authors in [57], conducted a series of experiments with convolitional neural networks trained on top of pre-trained word vectors for document classification tasks. They proved, that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks, suggesting that the pre-trained vectors are 'universal' feature extractors that can be utilized for various classification tasks. Learning task-specific vectors through fine-tuning results in further improvements. The vectors used were trained by [78] on 100 billion words of Google News, and are publicly available. Initially they keept the word vectors static and learn only the other parameters of the model. Additionally they proposed a simple modification to the architecture to allow for the use of both task-specific and static vectors. The CNN models discussed herein improve upon the state of the art on 4 out of 7 tasks, which include sentiment analysis and question classification. This work, is philosophically similar to [97], which showed that for image classification, feature extractors obtained from a pretrained deep learning model perform well on a variety of tasks—including tasks that are very different from the original task for which the feature extractors were trained. With the pre-trained word embeddings, neural networks demonstrate their great performance in many NLP tasks. Socher [111] used semi-supervised recursive autoencoders to predict the sentiment of a sentence. Their method learns vector space representations for multi-word phrases. In sentiment prediction tasks these representations outperform other state-of-the-art approaches on commonly used datasets, such as movie reviews, without using any pre-defined sentiment lexica or polarity shifting rules. they also evaluate the model's ability to predict sentiment distributions on a new dataset based on confessions from the experience project. The dataset consists of personal user stories annotated with multiple labels which, when aggregated, form a multinomial distribution that captures emotional reactions. Addiotionally, [101] introduced a method for paraphrase detection based on recursive autoencoders (RAE). Their unsupervised RAEs are based on a novel unfolding objective and learn feature vectors for phrases in syntactic trees. These features are used to measure the word- and phrase-wise similarity between two sentences. Since sentences may be of arbitrary length, the resulting matrix of similarity measures is of variable size. A novel dynamic pooling layer was indroduced which computes a fixed-sized representation from the variable-sized matrices. The pooled representation is then used as input to a classifier. The method outperforms other state-of-the-art approaches on the challenging MSRP paraphrase corpus.

5.5 Challenges

As we mentioned above sentiment analysis approaches aim to extract positive and negative sentiment bearing words from a text then classify the text as positive or negative or objective if no sentiment bearing words are appeared in the text. However, positive or a negative sentiment words may have opposite orientations in different domains. Moreover, a sentence containing sentiment words does not mean necessarily that express any sentiment. This can be seen frequently in questions or conditional sentences. Additionally, for successful analysis of sentiment, the opinion words should integrate with implicit data. The implicit data determine the actual behavior of sentiment words.

Ambiguity and vagueness have been considered as major issues since user reviews are often written using a loose style than standard texts, and often express sarcasm (mock or convey or irony), rhetoric or metaphor. Political discussion and extreme often include irony and sarcastic words; detection of such expression is a challenging task in opinion mining area. The fuzzy approach is quite useful to represent such feelings and expressions. Li and Tsai [66] framed a classification framework on the basis of fuzzy formal concept analysis (FFCA). For feature representation and dimensionality reduction, TF-IDF, Inverted Conformity Frequency (ICF) and Uniformity (Uni), and the relation between documents and terms have been considered using context matrices. The attribute lattice set was represented using terms of the document. The TF-IDF value represented the degree of significance of each term in the lattice object set, which were given to the classifier

Note that, not all languages have the same expressive power regarding sentiments. Quite recently, an important research part is working towards this issue. Although, high accuracies are not yet able to be produced, shedding light in unexplored corners of several approaches can definitely influence in the future re accuracy. Similarly, cross-domain sentiment analysis became a hot research problem to work upon. Cross-domain requires at least two domains: source domain on which a classifier is to be trained on, and target domain on which testing is to be performed. Work carried out in this area can be classified into two groups; the first group requires initial training set from source domain as well as from target domain. The learners in the second group of study are trained on source domain and tested on target domain.

Another significant problem is that companies are hiring fake reviewers to write fake reviews. Opinion spam detection and review usefulness measurement in general draw extremely attention recently. Most of the opinion spam detection techniques depends on three types of features related to a fake review, which includes content of review, meta-data of review, and real-life knowledge about the product. To begin with, the text can be analyzed using NLP and ML models in order to uncover whether a lie or deception is hidden in the text. Secondly, metadata of a review is worth checking. As metadata can be seen information as user-id, IP- address, geo-location etc. Last but not least, the world knowledge of the system plays important role in order a fake review to be recognized. For instance, a particular brand has very good reputation and an inferior brand has been shown as superior to that in some reviews posted during specific period, in such cases a review can be suspected as a fake review.

Thwarted expectations are a problem that is commonly faced in sentiment analysis and yet no effective methods have been generated to be dealt with. It is common for author to deliberately set up context only to refute it at the end.

All in all, since sentiment analysis is a sub-field of natural language processing it inhertis the challenges such as ambiguity, co-reference, Implicitness, inference etc. created hindrance in sentiment analysis too

Chapter 6

Experiments

6.1 Framework

Several tools exist for performing different machine learning tasks, such as classification, regression, and clustering. The tool used in this thesis was the Scikit Learn framework [92]. This is a Python framework built on NumPy [85], SciPy [53], and matplotlib [52]. Scikit Learn supports several classification methods, SVM, NB, MaxEnt, nearest neighbours, and random forest, to name a few. The main reason for choosing to use this framework for the task was the wide range of different classification methods, as well as a well-written and complementary framework documentation.The python library Keras [15] is the framework used for modelling some of the deep learning models used in this thesis. Keras is meant to be a minimalistic library with a focus on fast experimentation. It can be run on top of either TensorFlow or Theano, both enables running computations on GPU's. In this thesis the combination of Keras and Theano [2] is used. The python package Gensim [99] is used for handling the word vector models used. Keras, Theano and Gensim are currently only available as beta versions. The used versions are the developer versions 0.2.0 of Keras and 0.8.0 of Theano and the general beta version 0.12.1 of Gensim (these developer versions are updated from day to day without changing version number as they are bleeding edge). This implies algorithms could be changed and that the toolbox currently available is not written in stone.

6.2 IMDB dataset

The Stanford Large Movie Review (IMDB) Dataset [70] consists of 50,000" highly polar", binary labeled reviews from IMDB. These reviews are split 50:50 into training and testing sets. The distribution of labels within each subset of data is balanced. The dataset also includes a further 50,000 unlabeled reviews which may be used for unsupervised training. We found that reviews averaged 267.9 tokens in length with a standard deviation of 198.8 tokens; the precise distribution of review lengths is shown in Figure 6.1

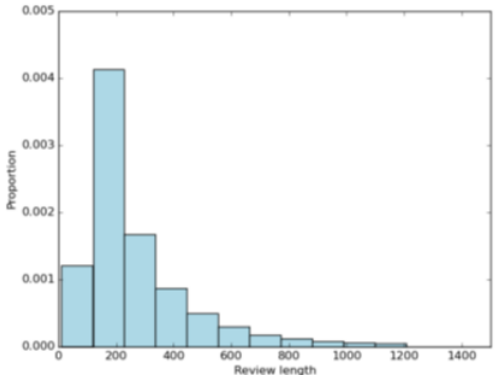


Figure 6.1: Distribution of review lengths in the IMDB dataset

6.3 Experiments with symbolic features representations

The reviews of IMDB dataset are processed in order noisy features to be removed. Then the textual data are transformed using vectorization techniques. In the final step, the vectors are fed to machine learning algorithms in order the sentiment classification to be performed. The step of the approach is shown in the following figure: 6.2

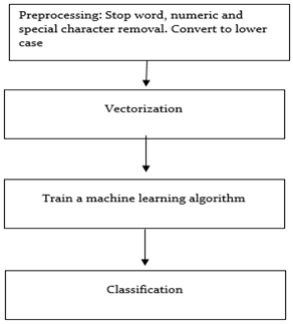


Figure 6.2: Diagrammatic view of the proposed approach

Step 1: In this step, we implement what we mentioned in section 4.1. Usually, the text of reviews sometimes consists of absurd data, that need to remove in order classification process not to be affected. We removed special and numeric characters and we convert all reviews into lowercase. Stop words are not removed.

Step 2: After the preprocessing of text reviews, they need to be converted to a matrix of numeric vectors. The following methodologies are considered for conversion of text to numeric vectors:

- Bag of word model
- TF/IDF model

Step 3: After the conversion of raw text into numeric vectors those vectors are fed to the following classifiers in order the determination of sentiment classification to be performed:

- Multinomial Naïve Bayes
- Support Vector Machine
- Logistic Regression

6.3.1 Bag of words features

Hyperparameter search is known to be a key challenge in applied machine learning. There are known procedures, such as grid search, but in practice, this problem is more difficult than it is in theory. The climax of this challenge is the explosive count of combinations of hyperparameters, where each combination could potentially outperform a very distant other combination, making it a non-convex problem.

Exhaustive optimization searches were done for the classification tasks, where several variations upon the input parameters were experimented with. This was done in order to attempt finding the optimal parameter setting for the classification algorithms. The optimal parameters for a classification task depends on the classifier used and the contextual aspects of the datasets on which it is used. Finding the optimal parameters for a task can greatly impact the performance of said task. All the parameter combinations used for optimization can be seen in table 6.2

Name	Range
N-gram range	1-1-1-2-1-3-1-4-2-2-2-3-3-3
Max DF	0.5 - 0.7 - 0.9 - 1.0
Min DF	0.2-0.3-0.4-0.5 -0.7 - 0.9 - 1.0
Alpha(NB-specific)	0.1 - 0.3 - 0.5 - 0.7 - 0.8 - 1.0
C(SVM-specific)	0.1 - 0.3 - 0.5 - 0.7 - 0.8 - 1.0
C(MaxEnt-specific)	Penalty(MaxEnt-specific) l1 - l2

Table 6.1: Parameter Combination For Optimization (BoW)

Name	Range
N-gram range	1-1 ,1-2 ,1-3 ,1,4, 2-2, 2-3, 3-3
Max DF	0.5 - 0.7 - 0.9 - 1.0
Min DF	0.2-0.3-0.4-0.5 - 0.7 - 0.9 - 1.0
Use IDF	True - False
Smooth IDF	True - False
Sublinear TF	True - False
Alpha(NB-specific)	0.1 - 0.3 - 0.5 - 0.7 - 0.8 - 1.0
C(SVM-specific)	0.1 - 0.3 - 0.5 - 0.7 - 0.8 - 1.0
C(MaxEnt-specific)	0.1 - 0.3 - 0.5 - 0.7 - 0.8 - 1.0
Penalty(MaxEnt-specific)	l1 - l2

Table 6.2: Parameter Combination For Optimization (TF/IDF)

The Parameters used for text vectorization with bow model are the range of N grams used as features, and the Max , Min document frequency for using the grams as features. More three parameters were used for TF-IDF vectoring; Use IDF, Smooth IDF, and Sublinear TF, all three of them Boolean values. Finally, 4 algorithm specific parameters were used. The Alpha parameter of the NB classifier, which is the Laplace/Lidstone smoothing weight. The C Parameter in the SVM, which influences the margin of the SVM hyperplane. And there is lastly two MaxEnt-specific parameters; the C and the penalty parameters

- Application of Multinomial Naïve Bayes classifier

Naïve Bayes was a certain choice to be involved in the project as a classifier, because it has been one of the popular machine learning methods for many years. The framework in Naïve Bays is attractive since its simplicity in various tasks and it offers reasonable performance, obtained in the tasks even though the learning is based on an unrealistic independence assumption.

Method	Confusion Matrix			Evaluation Parameter			Accuracy
Unigram		Corect labels		Precision	Recall	F-Measure	81.53%
		Positive	Negative				
	Positive	11008	1492	0.86	0.75	0.80	
	Negative	3126	9374	0.78	0.88	0.83	
Bigram		Corect labels		Precision	Recall	F-Measure	86.99%
		Positive	Negative				
	Positive	11209	1291	0.89	0.84	0.87	
	Negative	1962	10538	0.85	0.90	0.87	
Trigram		Corect labels		Precision	Recall	F-Measure	87.28%
		Positive	Negative				
	Positive	10986	1514	0.87	0.88	0.87	
	Negative	1665	10835	0.88	0.87	0.87	
Unigram+Bigram		Corect labels		Precision	Recall	F-Measure	85.72%
		Positive	Negative				
	Positive	11193	1307	0.89	0.82	0.85	
	Negative	2262	10238	0.83	0.90	0.86	
Bigram+Trigram		Corect labels		Precision	Recall	F-Measure	88.15%
		Positive	Negative				
	Positive	11200	1300	0.89	0.87	0.88	
	Negative	1663	10837	0.87	0.90	0.88	
Unigram+Bigram+Trigram				Precision	Recall	F-Measure	87.25%
		Positive	Negative				
	Positive	11204	1296	0.86	0.90	0.88	
	Negative	1892	10608	0.89	0.85	0.87	

Table 6.3: Various evaluation metrics for Multinomial Bayes Classifier for various n-gram range

As shown in the Table 6.3, the higher accuracy achieved is when the classifier is fed with trigram features. Note that, multinomial naïve Bayes is probabilistic methods where exist no dependency among features. In fact, when the analysis is carried out on higher level of n-grams features, it is shown that the classifier achieved better results because of the repetition of words that affects the probability distribution of the document. However, when the range of n-gram exceeds the trigram the accuracy is decreasing. For instance for n-gram range of $(1, 5)$ accuracy is 87.90

- Application of Logistic Regression Classifier

In Table 6.4 we observed that the accuracy of bigrams is higher than unigram and trigrams. However, the unigram and trigram methods when combined with bigram and between themselves, the accuracy values of various combinations are observed to be significantly higher. One can say that , when the sentiment bearing words appearing more than once leads to better classification results due to the probabilistic nature of logistic regression.

Method	Confusion Matrix			Evaluation Parameter			Accuracy
Unigram		Corect labels		Precision	Recall	F-Measure	85.34%
		Positive	Negative				
	Positive	10828	1672	0.86	0.75	0.80	
	Negative	1992	10508	0.78	0.88	0.83	
Bigram		Corect labels		Precision	Recall	F-Measure	87.73%
		Positive	Negative				
	Positive	10891	1609	0.87	0.88	0.88	
	Negative	1459	11041	0.88	0.87	0.88	
Trigram		Corect labels		Precision	Recall	F-Measure	84.16%
		Positive	Negative				
	Positive	10348	2152	0.83	0.86	0.84	
	Negative	1807	10693	0.85	0.83	0.84	
Unigram+Bigram		Corect labels		Precision	Recall	F-Measure	89.28%
		Positive	Negative				
	Positive	11156	1344	0.89	0.89	0.89	
	Negative	1337	11163	0.89	0.89	0.89	
Bigram+Trigram		Corect labels		Precision	Recall	F-Measure	88.16%
		Positive	Negative				
	Positive			0.88	0.89	0.88	
	Negative			0.89	0.87	0.88	
Unigram+Bigram+Trigram				Precision	Recall	F-Measure	89.56%
		Positive	Negative				
	Positive	11178	1322	0.89	0.90	0.90	
	Negative	1289	11211	0.90	0.89	0.90	

Table 6.4: Various evaluation metrics for Logistic Regression Classifier for various n-gram range

- Linear Support Vector Machine

Comparing the Linear SVMs results with those of logistic regression one can find some interesting observations. In linear SVM when unigrams are considered better results are yield whereas in logistic regression are the worst. As SVM method is a non-probabilistic linear classifier and trains model to find hyperplane in order to separate the dataset, the unigram model which analyzes single words for analysis gives better result. In bigram and trigram, there exists multiple word combinations, which, when plotted in a particular hyperplane, confuses the classifier and thus, it provides a less accurate result in comparison with the value obtained using unigram. In fact, when trigrams and bigrams are combined with unigrams remarkable results are produced while bigrams and trigrams combination produces lower results than the unigrams alone.

Method	Confusion Matrix			Evaluation Parameter			Accuracy
Unigram		Corect labels		Precision	Recall	F-Measure	88.38%
		Positive	Negative				
	Positive	11021	1479	0.88	0.89	0.88	
	Negative	1425	11075	0.89	0.88	0.88	
Bigram		Corect labels		Precision	Recall	F-Measure	86.78%
		Positive	Negative				
	Positive	10665	1835	0.86	0.88	0.87	
	Negative	1469	11031	0.88	0.85	0.87	
Trigram		Corect labels		Precision	Recall	F-Measure	83.86%
		Positive	Negative				
	Positive			0.83	0.85	0.84	
	Negative			0.85	0.82	0.84	
Unigram+Bigram		Corect labels		Precision	Recall	F-Measure	89.60%
		Positive	Negative				
	Positive	11138	1362	0.90	0.89	0.90	
	Negative	1238	11262	0.89	0.90	0.90	
Bigram+Trigram		Corect labels		Precision	Recall	F-Measure	87.23%
		Positive	Negative				
	Positive			0.86	0.89	0.87	
	Negative			0.88	0.86	0.87	
Unigram+Bigram+Trigram				Precision	Recall	F-Measure	89.64%
		Positive	Negative				
	Positive			0.89	0.90	0.90	
	Negative			0.90	0.88	0.90	

Table 6.5: Various evaluation metrics for Linear Support Vector Machine for various n-gram range

6.3.2 Term frequency & inverse document frequency

In the folowing Section we present the results for our three classifiers using TF-IDF features

- Application of Multinomial Naïve Bayes classifier

Method	Confusion Matrix			Evaluation Parameter			Accuracy
Unigram		Corect labels		Precision	Recall	F-Measure	83.55%
		Positive	Negative				
	Positive	11113	1387	0.88	0.78	0.83	
	Negative	2726	9774	0.80	0.89	0.84	
Bigram		Corect labels		Precision	Recall	F-Measure	88.17%
		Positive	Negative				
	Positive	11347	1153	0.90	0.86	0.88	
	Negative	1805	10695	0.86	0.91	0.88	
Trigram		Corect labels		Precision	Recall	F-Measure	86.98%
		Positive	Negative				
	Positive	10968	1532	0.88	0.86	0.87	
	Negative	1723	10777	0.86	0.88	0.87	
Unigram+Bigram		Corect labels		Precision	Recall	F-Measure	87.30%
		Positive	Negative				
	Positive	10782	1718		0.87	0.89	
	0.88	1352	11148	0.89	0.86	0.88	
Bigram+Trigram		Corect labels		Precision	Recall	F-Measure	88.78%
		Positive	Negative				
	Positive	11308	1192	0.90	0.87	0.88	
	Negative	1666	10834	0.87	0.90	0.89	
Unigram+Bigram+Trigram				Precision	Recall	F-Measure	88.17%
		Positive	Negative				
	Positive	11331	1169	0.90	0.86	0.88	
	Negative	1789	10711	0.86	0.91	0.88	

Table 6.6: Various evaluation metrics for Multinomial Bayes Classifier for various n-gram ranges(TF-IDF)

Again , the Multinomial Naïve Bayes results are very competitive compared to the states of the art classifiers. The maximum accuracy achieved is 88.78%

- Application of Linear Support Vector Machine

Method	Confusion Matrix			Evaluation Parameter			Accuracy
Unigram		Corect labels		Precision	Recall	F-Measure	88.98%
		Positive	Negative				
	Positive	11109	1391	0.89	0.89	0.89	
	Negative	1364	11136	0.89	0.89	0.89	
Bigram		Corect labels		Precision	Recall	F-Measure	87.88%
		Positive	Negative				
	Positive	10802	1698	0.87	0.89	0.88	
	Negative	1331	11169	0.89	0.86	0.88	
Trigram		Corect labels		Precision	Recall	F-Measure	84.85%
		Positive	Negative				
	Positive	10332	2168	0.83	0.87	0.85	
	Negative	1619	10881	0.86	0.83	0.85	
Unigram+Bigram		Corect labels		Precision	Recall	F-Measure	89.74%
		Positive	Negative				
	Positive	11158	1342	0.89	0.90	0.90	
	0.88	1223	11277	0.90	0.89	0.90	
Bigram+Trigram		Corect labels		Precision	Recall	F-Measure	89.96
		Positive	Negative				
	Positive	11163	1337	0.89	0.89	0.90	
	Negative	1172	11328	0.90	0.91	0.90	
Unigram+Bigram+Trigram				Precision	Recall	F-Measure	90.53%
		Positive	Negative				
	Positive	11334	1166	0.91	0.90	0.90	
	Negative	1207	11293	0.90	0.91	0.91	

Table 6.7: Various evaluation metrics for for Linear Support Vector Machine for various n-gram ranges(TF-IDF)

As it was expected the Support Vector machine classifier 6.7 performed exceptional and in this method. The Logistic Regression classifier produced slightly worse results compared to SVM where achieved maxim accuracy of 89.68% almost 1% lower than SVMs.

6.3.3 Discussion

We used three state-of-the-art classifiers namely Naive Bayes(NB), Logistic Regression (LR) and Support Vector Machines(SVM) together with two feature selection methods. The results of machine learning-based classifiers incorporating N-Gram Bag of-Words features with N ranging from 1(unigram) to 3 are summarized in Tables 6.3 , 6.4 and 6.5. Generally, machine learning classifiers achieved very inspiring results in evaluation. All of four measurements are very high compared to the state-of-results of IMDB database (91.2%). Naive Bayes is one of the simplest classifier yet it archived at least $0.8153 - 0.8815$ in average accuracy. The logistic regression achieved almost 1.5 better accuracy than the naïve Bayes as it was expected. In general, that classifier achieved remarkable results were achieved at least $0.8416 - 0.8956$ in average accuracy. Furthermore, Linear SVM it was excepted to perform slightly better in terms of accuracy compared to linear Regression. The best accuracy SVM achieved was only 0.07 better of logistic Regression but the lowest

Parameters	NB	SVM	LR
Accuracy	88.15%	89.64%	89.56%
N-gram range	2-3	1-3	1-3
Min DF	0.3	0.3	0.3
Max DF	0.9	0.9	0.9
Alpha	1.0	1.0	1.0
C(SVM)	1.0	1.0	1.0
C(MaxEnt)	1.0	1.0	1.0
Penalty	-	L2	L2

Table 6.8: Parameter values with best performance in Sentiment Classification (BoW)

Parameters	NB	SVM	LR
Accuracy	88.78%	90.74%	89.68%
N-gram range	2-3	1-3	2-3
Min DF	0.3	0.3	0.3
Max DF	0.9	0.9	0.9
Use IDF	True	True	True
Smooth IDF	True	True	True
Sublinear TF	True	True	True
Alpha	0.3	0.3	0.3
C(SVM)	10	10	10
C(MaxEnt)	1.0	1.0	
Penalty	-	L2	L2

Table 6.9: Parameter values with best performance in Sentiment Classification (TF-IDF)

accuracy was 1 lower than the Logistic Regression. Based on our observations ,classifiers achieved better results when n-grams units where combined together due to the fact that single units of n-grams affects the classification process due to the nature of classifiers. For instance, SVM achieved its lowest accuracy when trigrams were used because in trigrams exist multiply combinations thus the right hyperplane cannot be found easily as when unigrams are used. In terms of average performance , combination of Unigram+Bigram+Trigram definitely dominated all other n-grams features and their combinations.

The overall performance of all three classifiers when TF-IDF features were used was significantly better than the simple bag-of-words as shown in Tables 6.6 , 6.6. The lowest accuracy achieved was 2 higher and the best accuracy achieved was 1% higher than the simple BoW. Taking into consideration the limitations of those approaches and their simplicity the remarkable 90.53% achieved was not expected. In tables 6.8 , 6.9 best parameters and best results are shown.

6.4 Word Embeddings

6.4.1 Word2Vec method work flow

The following figure summarizes the step we follow to produce the results that are presented in the following subsections

In the next section we describe in depth the above steps.

6.4.2 Word2Vec hyperparameter optimization

We trained the Word2Vec by counting the co-occurrence of words in the same sentence and then we created a vector representation for each word then the matrix is fed to a classifier in order our

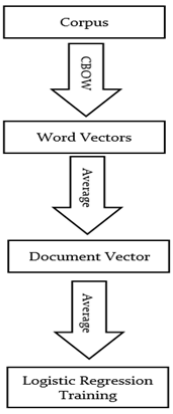


Figure 6.3: Workflow Summary of Document Classification using Word2Vec features

Name	Range	Best choice
Minimum word count	[5,100]	40
Epochs	[5,50]	25
Word Vector Dimensionality	[100,5000]	500
Window size	[5,100]	45
Negative Sampling	[5,25]	25
Frequent Word Downsampling	[10-10, 0.1]	0.001
Hierarchical Softmax	[True, False]	False

Table 6.10: The value ranges for the most important hyperparameters of Word2Vec model

prediction models to be trained. During the training phase a lot of parameters should be taken in account in order better results to be produced:

- Architecture: Architecture options are skip-gram (default) or continuous bag of words. We found that Cbow performs better and is much faster than Skip-gram
- Training algorithm: Hierarchical Softmax (default) or negative sampling. For us, the default worked well. Downsampling of frequent words: The Google documentation recommends values between .00001 and .001. For us, values closer 0.001 seemed to improve the accuracy of the final mode.
- Word vector dimensionality: More features result in longer runtimes, and often, but not always, result in better models. Reasonable values can be in the tens to hundreds;300 ,500 worked best for us where we did a complete search for the rest hyperparameters.
- Context / window size: How many words of context should the training algorithm take into account? We used a wide range from 4 to 100 45 worked well for us
- Minimum word count: This helps limit the size of the vocabulary to meaningful words. Any word that does not occur at least this many times across all documents is ignored. Reasonable values could be between 10 and 100. In this case, since each movie occurs 30 times, we set the minimum word count to 40, to avoid attaching too much importance to individual movie titles.

Note that an advantage of word2vec is that we can train the model without having labeled data, which enhances our training set to 50,000 reviews. To make predictions on the test data, we experiment multiple ways of coming up with a vector representation of a review:

- A naïve but effective way of doing so is to represent each review as the average of the word vectors representing it
- Another way to so was by first clustering the words from the word2vec model using K-Means and then assigning each cluster an index. In our experiments this method led to 3-12% lower results thus is not furthermore considered.

6.4.3 From words to paragraphs: vector averaging

Method	Confusion Matrix			Evaluation Parameter			Accuracy
100		Corect labels		Precision	Recall	F-Measure	86.41%
		Positive	Negative				
	Positive			0.86	0.87	0.86	
	Negative			0.87	0.86	0.86	
200		Corect labels		Precision	Recall	F-Measure	86.39%
		Positive	Negative				
	Positive	10761	1739	0.86	0.87	0.86	
	Negative	1680	10820	0.86	0.86	0.86	
300		Corect labels		Precision	Recall	F-Measure	88.51%
		Positive	Negative				
	Positive	11049	1451	0.88	0.88	0.88	
	Negative	1513	10987	0.88	0.88	0.88	
400		Corect labels		Precision	Recall	F-Measure	87.48%
		Positive	Negative				
	Positive	10924	1576	0.87	0.88	0.87	
	0.88	1554	10946	0.88	0.87	0.87	
500		Corect labels		Precision	Recall	F-Measure	88.73%
		Positive	Negative				
	Positive	11091	1409	0.89	0.88	0.89	
	Negative	1442	11058	0.88	0.89	0.89	
600				Precision	Recall	F-Measure	87.78%
		Positive	Negative				
	Positive	10967	1533	0.88	0.88	0.88	
	Negative	1522	10978	0.88	0.88	0.88	

Table 6.11: Document features Word2Vec and Logistic Regression

One challenge with the IMDB dataset is the variable-length reviews. We need to find a way to take individual word vectors and transform them into a feature set that is the same length for every review.

Since each word is a vector in N-dimensional space, we can use vector operations to combine the words in each review. One method we tried was to simply average the word vectors in a given review (for this purpose, we removed stop words, which would just add noise). The results for different dimensions are given in Table 6.11.

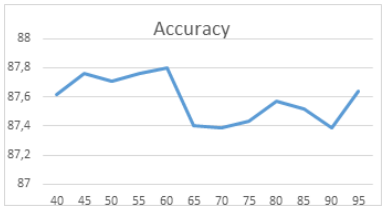


Figure 6.4: Accuracy Vs Minimum Word Count (window size=10, size=300)

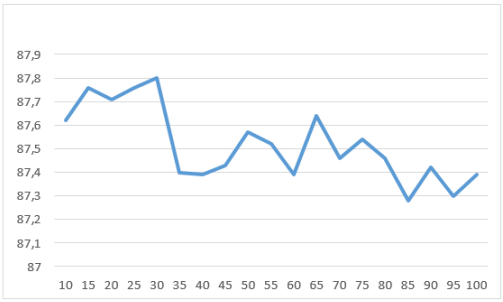


Figure 6.5: Accuracy Vs Window Size (size=300,min_count=40)

In figure 6.4 we examine how minimum word count and window size affects the overall accuracy of the classification task.

Remove the words that are less frequent than min word count can affect the accuracy as the figure shows 6.4 . In fact, the maximum accuracy is achieved when the minimum counts have value of 60 and then the accuracy drops significantly.

6.4.4 Doc2Vec hyperparameter optimization

Here, we implement the Paragraph Vectors model as described in [63]. This model is similar to the word2vec model, but with the difference that each sentence (or paragraph) also has a vector representation (in addition to the words in the sentence). When it comes to training and prediction, it treats each sentence as one vector by concatenating the paragraph vector with the word vectors (Dm_concat) or averaging those vectors (Dm_mean) and also there is PV-DBOW model which is equivalent to skip-gram model (cbow=0). The parameters of doc2vec mirrored those from word2vec and we used Logistic Regression as achieved best results compared to Linear SVM.

We wasn't able to reproduce the results presented in Mikolov's paper however the maximum accuracy we achieved was 1.1 % less than the state of the art results. In table 6.13 we present the results using the three models mentioned above and the combination of those models. We used again Logistic Regression as a classifier. In figure 6.6 it is shown the difference in performance of the three models and the combinations of those. It is clear that, the PV-DBOW perform better compared to the other two PV-DM models. It is remarkable that DM with concatenate word vectors reached 86.70% and then dropped to 65% when the size of vectors was increased. Although number of epochs increases the overall quality of the vectors is up to 25 where after that the accuracy drops significantly. Note that, for larger size of vectors (1000) the time needed for 25 training epochs is about 18 hours where for 100 dimensions is about 2 hours. All in all, we weren't able to outperform the results obtained from TF-IDF. For Word2Vec we produced maximum accuracy of 88.73%. On the other hand, Doc2Vec produced results almost 1% higher than word2vec. In fact, it produced high accuracy value on low dimensions where the word2vec produced its maximum performance on 500 dimensions. We highly tuned the models but we weren't able to come close to results reported in [63].

Name	Range	Best choice
Minimum word count	[1,100]	2
Epochs	[5,50]	25
Word Vector Dimensionality	[100,5000]	100
Window size	[1,100]	10
Negative Sampling	[5,25]	5
Frequent Word Downsampling	[10-10, 0.0]	-
Hierarchical Softmax	[True, False]	False
DBOW	[True, False]	True
Dm_mean	[0,1]	0
Dm_concat	[0,1]	0

Table 6.12: The value ranges for the most important hyperparameters of Doc2Vec model

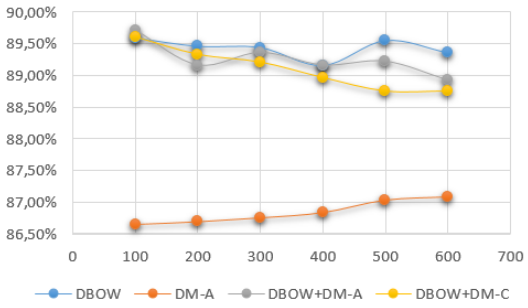


Figure 6.6: Accuracy vs Dimensions

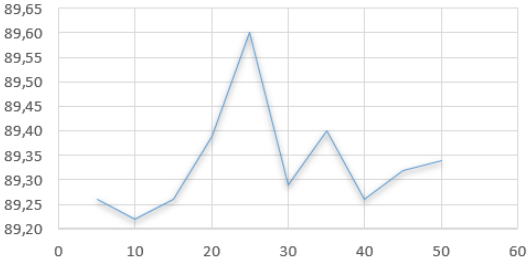


Figure 6.7: Accuracy Vs Epochs

Dimensions	Models Accuracy				
	DBOW	DM-A	DM-C	DBOW+DM-A	DBOW+DM-C
100	89,59%	86,66%	81,68%	89,71%	89,60%
200	89,46%	86,70%	86,70%	89,16%	89,33%
300	89,44%	86,76%	67,20%	89,36%	89,21%
400	89,16%	86,84%	65,31%	89,16%	88,97%
500	89,55%	87,03%	65,33%	89,22%	88,76%
600	89,36%	87,08%	65,64%	88,92%	88,76%

Table 6.13: Classification Accuracy using Do2Vec Features and Logistic Regression

6.4.5 Discussion

After our experiments is apparent that Doc2vec outperforms Word2vec averaging method. However, the difference is less than 1% in terms of accuracy. We experiment with both variants of doc2vec (dbow and dmpv) and word2vec (skip-gram and cbow). Cbow produced the best results and was faster compared to the skip-gram model. The runtime of cbow was the half of skip-gram model. Additionally, for doc2vec models we found that cbow outperforms the Dm model. While training the doc2vec models we noticed that the model over fits easily. In order one to address this problem, should shuffle the train data in each training epoch. Although, doc2vec produces better results the training phase is very slow and challenging due to the fact that the model overfits on the paragraph labels. For both models, the hyper-parameters that affect the overall performance the most, is the dimension of the vectors, the minimum word count and the window size.We have proven that, while the values of those parameters are increased the accuracy is increased also but up to a point. However, when the dimensions are increased the training time of the models is sometimes inefficient. For instance, vectors of 5000 size needed 23 hours to be trained with word2vec and 36 for doc2vec. A parameter that one should be carefull is the subsampling parameters. We performed exaustive reseach and we found that whatever value it takes affects negative the accuracy of doc2vec model so we exclude this parameter while we were training the model. Additionally, we have focused on quantitative evaluation of doc2vec and word2vec. The qualitative difference between doc2vec and word2vec document embeddings, however, remains unclear.

6.5 Deeply learned distributed representations of features

6.5.1 Convolution neural networks

We experimented with CNN on sentiment classification task. The network consists of embedding layer followed by a convolution layer connected to a max pooling layer. Additionally, we added a hidden layer between the max pooling layer and the output layer. We used ReLU activation function and minimized square loss with L2 regularization by the use of Adam optimizer. We only used the 5000 words that appeared most frequently in the training set; Out-of-vocabulary words were represented by a zero vector.

While neural networks have a large capacity to learn complex decision functions they tend to easily overfit especially on small and medium sized datasets. To mitigate the overfitting issue, we augment the cost function with l2-norm regularization terms for the parameters of the network. We also use another popular and effective technique to improve regularization of the neural networks — dropout [112]. Dropout prevents feature co-adaptation by setting to zero (dropping out) a portion of hidden units during the forward phase when computing the activations at the sigmoid output layer.

The results was better than word2vec and doc2vec. The maximum accuracy achieved was 90.17%. Table 6.15 summarize the results. In figures 6.8a , 6.8 the learning curve ingredients are shown. Although we used dropout layer, it was inevitable the model to overfit as the epochs were

Name	Range	Best Choice
Maximum number of features	[1000,100000]	5000
Review Length	[100,800]	200
Word Vector Size	[30,500]	50
Batch size	[32,512]	32
Window size	[3-21]]	6
Epochs	[1,10]	7

Table 6.14: Hyperparameters for CNN Model

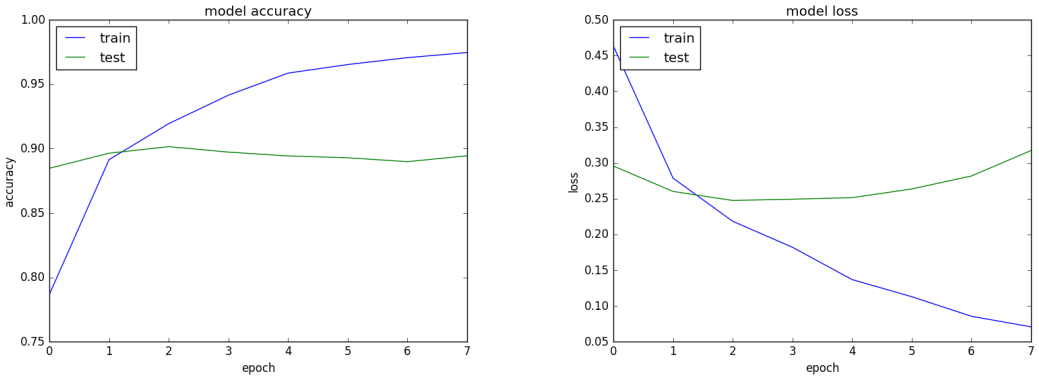
Batch Size	Accuracy
32	90.17%
64	89.70%
96	88.60%
128	88.45%

Table 6.15: Classification Accuracy for Diffeernt values of batch size

increased leading to increase of loss and decrease of accuracy while the train accuracy reached 97% . The results achieved were competitive with the results published in literature. However, a new trend in convolution neural networks concerning sentiment analysis is to omit embedding layer by using pretrained vector produced by Word2vec and Doc2vec methods. We tried, different setups but we werent able to achieved compatitive results.

6.5.2 Long short term memory

Since the documents are long, one might expect that it is difficult for recurrent networks to learn. We however find that with tuning, it is possible to train LSTM recurrent networks to fit the training set. Our one-layer LSTM with random initialized word vectors ,performed best with word vector and hidden dimensions of 200. However, we tested dimensions ranging from 50 to 250; overall, the LSTM easily attains high accuracy on the training set while failing to generalize to the test set. Optimal performance was obtained after 10-15 training epochs. The results are shown in Table 6.16



(a) CNN train and validation accuracy

Figure 6.8: CNN model loss function

Dimensions	Accuracy
50	88.09%
100	88.44%
150	88.83%
200	89.90%
250	88.66%

Table 6.16: Acurracy of LSTM network for different embedding size

For our best results we used an LSTM 1 layer deep, with word and hidden vector dimensions of 200, and dropout regularization with a drop rate of 0.5. Updates were applied using Adam with an initial learning rate of 1, decay of 0.99 and epsilon of 1e-8.

6.5.3 Bidirectional long short term memory

A huge drawback of the standard RNN is that post-word context is not considered sufficiently as the document is observed only in one direction. However, in order to determine sentiment, the direction should matter and therefore, a bidirectional long short term memory is implemented: Accumulating in two directions rather than one doubles the number of parameters and allows for more flexibility. Here, the model runs through the sequence in reverse order with a different set of parameters that have to be updated. Note that, to specify the backward channel, we just need to invert the sequence of words and perform the same LSTM as we did before, on the other direction. Table 6.17 summarises the results.

Dimensions	Accuracy
50	87.12%
100	87.78%
150	87.88%
200	87.68%
250	86.64%

Table 6.17: Acurracy of Bi-LSTM network for different embedding size

For our best results using Bi-LSTM we used an LSTM 1 layer deep, with word and hidden vector dimensions of 150, and dropout regularization with a drop rate of 0.5. Updates were applied using Adam with an initial learning rate of 1, decay of 0.99 and epsilon of 1e-8.

6.5.4 Combination of CNN and LSTM

Convolutional neural network (CNN) and recurrent neural network (RNN) are two mainstream architectures for sentiment analysis which adopt totally different ways of understanding natural languages. In this section, we combine the strengths of both architectures for natural language representation and document classification. C-LSTM utilizes CNN to extract a sequence of higher-level phrase representations, and are fed into a long short-term memory recurrent neural network (LSTM) to obtain the document representation. C-LSTM is able to capture both local features of phrases as well as global and temporal sentence semantics. The experimental results show that the C-LSTM does not produce better results than the combined architectures. This might be caused because of the long length of the reviews are examined. In shorter texts, it has been shown that this architecture produces remarkable results.We found that the model is higly sensitive to the input lenght , lenght of pullin filter. Additionally , the model achieves high accuracy on train set after only 2 epochs. We trained the model in epoch range (2,5). The results are shown in Table 6.18.

Dimensions	Accuracy
50	87.12%
100	87.19%
150	87.36%
200	86.41%
250	87.22%

Table 6.18: CNN-LSTM accuracy results

6.5.5 Discussion

In the previous sections we compared with CNN, RNN models. Empirical experiments show that CNN models usually yield better performance than RNNs which include LSTM and Bi-LSTM. It may be caused by RNN based models, especially the LSTMs, are hard to be trained, these models are really sensitive to the hyper parameters and latter words make more influence on the final text representation than former words in RNN models. CNN models usually perform remarkably well on many NLP and IR tasks ([49, 57, 61]). CNN based models use a fixed window of words as contextual information and the performance of a CNN is influenced by the window size. A small window may result in a loss of some long-distance patterns, whereas large windows will lead to data sparsity ([19, 61]). However, CNNs outperform the RNN and the CNN-LSTM model, we know that convolution structure can capture local context information, and recurrent can capture global information. We consider that in CNN-LSTM model, it does not make sense to use convolution layer and max-pooling layer before recurrent layer. Such architecture means using local information features extracted from CNN as RNN's input, but local information features do not have sequence relationship.

Chapter 7

Conclusion and future work

In this thesis, we studied a wide range of NLP classification models. Our investigations consisted of two main parts. In the first part, we used the Large movie dataset provided by [70] and applied the bag of words, TF/IDF, word2vec, doc2vec models to represent words numerically. We then used several classifiers, including logistic regression, support vector machine, and Multinomial naïve bays to perform the binary classification task. When we use these classifiers, one of the challenges is to aggregate word vectors into a single feature vector for each review. We tried vector averaging and clustering to produce the aggregated feature vectors. As the Table 7.1 shows, the traditional feature extraction methods despite their simplicity still performs remarkably, where BOW-SVM achieved maximum accuracy of 89.64% and SVM-TF-IDF 90.53%. The deep learning inspired methods, doc2vev and word2vec produced similar or outperformed the simple bag of words model. However, while Le and Mikolov [63] report state-of-the art results over sentiment analysis task in large movie reviews dataset using doc2vec, others (including the second author of the original paper in follow-up work) have struggled to replicate this result. Given this background of uncertainty regarding the true effectiveness of doc2vec and confusion about performance differences between dbow and dmpv, we tried to shed light on a number of empirical questions: which is better out of dmpv and dbow? is it possible to improve doc2vec through careful hyper-parameter optimization? To this end, we present a formal and rigorous evaluation of doc2vec. Our findings reveal that dbow, despite being the simpler model, is superior to dmpv. When trained over large external corpora, hyper-parameter tuning, we find that doc2vec performs very strongly compared to both a word embedding averaging (word2vec) and to our BOW baseline. Doc2vec outperforms word2vec embeddings with significant difference, almost 1%. Additionally , we performed numerous experiments and for word2vec model. We found that Cbow outperforms Skip-gram model in terms of accuracy. Training skip-gram model is by far more time consuming task compared to the training time of Cbow.

However, the above described models, except doc2vec, are suffering from losing the order of words in documents. This motivated the second part of our work, where we implemented the recurrent and convolutional neural networks to train a binary sentiment analyzer. When we compare neural network approaches (RNN, CNN) to our simple baseline model (i.e BOW-SVM) the experimental results show that the neural network approaches outperform the traditional methods. It proves that neural network based approach can effective compose the semantic representation of texts. Neural networks can capture more contextual information of features compared with traditional methods based on Bow model, and may suffer from the data sparsity problem less. When comparing CNNs to RNNs, we can see that the convolution-based approaches achieve better results. This illustrates that the convolution-based framework is more suitable for constructing the semantic representation of texts compared with previous neural networks. We believe the main reason is that CNN can select more discriminative features through the max-pooling layer and capture contextual information through convolutional layer.

We also compare RNN variations, such as LSTM and Bi- LSTM to the CNN and find that the CNN outperforms the RNN in all cases. We believe that the window-based structure of CNNs captures contextual information better than recurrent structure in RNNs. However, when we combined those models , the results was lower that the worst RNN model. We consider that in CNN-LSTM model, it does not make sense to use convolution layer and max-pooling layer before

Method	Accuracy
SVM-BOW	89.64%
SVM-TF/IDF	90.53%
SVM-WORD2VEC	88.73%
lR-DOC2VEC	89.71%
CNN	90.17%
LSTM	89.90%
Bi-LSTM	87.88%
CNN+LSTM	87.22%
state-of-the-art [125]	91.22%

Figure 7.1: Comparative result of values on "Accuracy" result obtained of the experiments conducted in this Thesis

recurrent layer. Such architecture means using local information features extracted from CNN as RNN's input, but local information features do not have sequence relationship.

All in all , despite the loss of semantic information, bag-of-ngram based methods still achieve state-of-the-art results for tasks such as sentiment classification of long movie reviews. Many document embeddings methods have been proposed to capture semantics, but they still can't outperform bag-of-n gram based methods on this task.

References

[1] Ahmed Abbasi, Hsinchun Chen, and Arab Salem. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. ACM Trans. Inf. Syst., 26(3): 12:1–12:34, June 2008. ISSN 1046-8188. doi: 10.1145/1361684.1361685. URL http://doi.acm.org/10.1145/1361684.1361685.

[2] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, and Nicolas Ballas a. Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints, abs/1605.02688, May 2016. URL http://arxiv.org/abs/1605.02688.

[3] Ethem Alpaydin. Introduction to Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, 2004. ISBN 0262012111.

[4] Farah Benamara, Carmine Cesarano, Antonio Picariello, Diego Reforgiato, and V. S. Subrahmanian. Sentiment analysis: Adjectives and adverbs are better than adjectives alone. In Proceedings of the International Conference on Weblogs and Social Media (ICWSM), 2007. Short paper.

[5] Ian Goodfellow Yoshua Bengio and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016. URL http://www.deeplearningbook.org.

[6] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. Trans. Neur. Netw., 5(2):157–166, March 1994. ISSN 1045-9227. doi: 10.1109/72.279181. URL http://dx.doi.org/10.1109/72.279181.

[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. J. Mach. Learn. Res., 3:1137–1155, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944966.

[8] Yoshua Bengio, Yann Lecun, DeÌpartement Dâinformatique Et Recherche OpeÌrationnelle, Universite Ì De MontreÌal, L. Bottou, O. Chapelle, D. Decoste, and J. Weston (eds. Scaling learning algorithms towards ai, 2007.

[9] Yoshua Bengio, Olivier Delalleau, and Clarence Simard. Decision trees do not generalize to new variations. Computational Intelligence, 26(4):449–467, 2010. ISSN 1467-8640. doi: 10.1111/j.1467-8640.2010.00366.x. URL http://dx.doi.org/10.1111/j.1467-8640.2010.00366.x.

[10] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. CoRR, abs/1206.5538, 2012. URL http://arxiv.org/abs/1206.5538.

[11] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. 2012.

[12] Maureen Caudill. Neural networks primer, part i. AI Expert, 2(12):46–52, December 1987. ISSN 0888-3785. URL http://dl.acm.org/citation.cfm?id=38292.38295.

[13] Eugene Charniak. Statistical Language Learning. MIT Press, Cambridge, MA, USA, 1994. ISBN 0262032163.

[14] Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. Identifying sources of opinions with conditional random fields and extraction patterns. In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05, pages 355–362, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220620. URL http://dx.doi.org/10.3115/1220575.1220620.

[15] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[16] Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surface of multilayer networks. CoRR, abs/1412.0233, 2014. URL http://arxiv.org/abs/1412.0233.

[17] Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surface of multilayer networks. CoRR, abs/1412.0233, 2014. URL http://arxiv.org/abs/1412.0233.

[18] Hinrich Christopher D Manning, Prabhakar Raghavan. ntroduction to Information Retrieval. Cambridge University Press, 2008.

[19] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

[20] William B. Claster, Malcolm Cooper, and Philip@inproceedingsClaster:2010:TTC:1908634.1909343, author = Claster, William B. and Cooper, Malcolm and Sallis, Philip, title = Thailand – Tourism and Conflict: Modeling Sentiment from Twitter Tweets Using NaïVe Bayes and Unsupervised Artificial Neural Nets, booktitle = Proceedings of the 2010 Second International Conference on Computational Intelligence, Modelling and Simulation, series = CIMSIM '10, year = 2010, isbn = 978-0-7695-4262-1, pages = 89–94, numpages = 6, url = http://dx.doi.org/10.1109/CIMSiM.2010.98, doi = 10.1109/CIMSiM.2010.98, acmid = 1909343, publisher = IEEE Computer Society, address = Washington, DC, USA, keywords = Tourism, Sentiment Mining, SOM, Twitter, Social Networks, Semantic Web, Text Mining, Sallis. Thailand – tourism and conflict: Modeling sentiment from twitter tweets using naïve bayes and unsupervised artificial neural nets. In Proceedings of the 2010 Second International Conference on Computational Intelligence, Modelling and Simulation, CIMSIM '10, pages 89–94, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4262-1. doi: 10.1109/CIMSiM.2010.98. URL http://dx.doi.org/10.1109/CIMSiM.2010.98.

[21] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. URL http://doi.acm.org/10.1145/1390156.1390177.

[22] Corinna Cortes and Vladimir Vapnik. Support-vector networks. Machine Learning, 20(3): 273–297, 1995. ISSN 1573-0565. doi: 10.1023/A:1022627411411. URL http://dx.doi.org/10.1023/A:1022627411411.

[23] Bruce Croft, Donald Metzler, and Trevor Strohman. Search Engines: Information Retrieval in Practice. Addison-Wesley Publishing Company, USA, 1st edition, 2009. ISBN 0136072240, 9780136072249.

[24] Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. CoRR, abs/1511.01432, 2015. URL http://arxiv.org/abs/1511.01432.

[25] Y. Dang, Y. Zhang, and H. Chen. A lexicon-enhanced method for sentiment classification: An experiment on online product reviews. IEEE Intelligent Systems, 25(4):46–53, July 2010. ISSN 1541-1672. doi: 10.1109/MIS.2009.105.

[26] S. R. Das and M. Y. Chen. Yahoo! for amazon: Sentiment parsing from small talk on the web. In EFA 2001 Barcelona Meetings, 2001.

[27] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In Proceedings of the 12th International Conference on World Wide Web, WWW '03, pages 519–528, New York, NY, USA, 2003. ACM. ISBN 1-58113-680-3. doi: 10.1145/775152.775226. URL http://doi.acm.org/10.1145/775152.775226.

[28] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res., 12:2121–2159, July 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2021068.

[29] Charlotta Engström. Topic dependence in sentiment classification. Master's thesis, University of Cambridge, 2004.

[30] Ronen Feldman. Techniques and applications for sentiment analysis. Commun. ACM, 56(4): 82–89, April 2013. ISSN 0001-0782. doi: 10.1145/2436256.2436274. URL http://doi.acm.org/10.1145/2436256.2436274.

[31] Zoubin Ghahramani. Unsupervised learning. In Advanced Lectures on Machine Learning, pages 72–112. Springer-Verlag, 2004.

[32] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010), volume 9, pages 249–256, May 2010.

[33] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey J. Gordon and David B. Dunson, editors, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11), volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011. URL http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf.

[34] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey J. Gordon and David B. Dunson, editors, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11), volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011. URL http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf.

[35] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. CoRR, abs/1402.3722, 2014. URL http://arxiv.org/abs/1402.3722.

[36] Alex Graves. Supervised sequence labelling with recurrent neural networks. PhD thesis, Technical University Munich, 2008.

[37] Alex Graves. Generating sequences with recurrent neural networks. CoRR, abs/1308.0850, 2013. URL http://arxiv.org/abs/1308.0850.

[38] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. IEEE Trans. Pattern Anal. Mach. Intell., 31(5):855–868, May 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.137. URL http://dx.doi.org/10.1109/TPAMI.2008.137.

[39] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. CoRR, abs/1303.5778, 2013. URL http://arxiv.org/abs/1303.5778.

[40] Barbara Hammer. On the approximation capability of recurrent neural networks. Neurocomputing, 31(1â4):107 – 123, 2000. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/S0925-2312(99)00174-5. URL http://www.sciencedirect.com/science/article/pii/S0925231299001745.

[41] Zellig S. Harris. Distributional Structure, pages 3–22. Springer Netherlands, Dordrecht, 1981. ISBN 978-94-009-8467-7. doi: 10.1007/978-94-009-8467-7_1. URL http://dx.doi.org/10.1007/978-94-009-8467-7_1.

[42] Johan Håstad. Computational Limitations of Small-depth Circuits. MIT Press, Cambridge, MA, USA, 1987. ISBN 0262081679.

[43] Vasileios Hatzivassiloglou and Kathleen R. McKeown. Predicting the semantic orientation of adjectives. In Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics, EACL '97, pages 174–181, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics. doi: 10.3115/979617.979640. URL http://dx.doi.org/10.3115/979617.979640.

[44] S.S. Haykin. Neural Networks and Learning Machines. Number τ. 10 in Neural networks and learning machines. Prentice Hall, 2009. ISBN 9780131471399. URL https://books.google.gr/books?id=K7P36lKzI_QC.

[45] Bas Heerschop, Frank Goossen, Alexander Hogenboom, Flavius Frasincar, Uzay Kaymak, and Franciska de Jong. Polarity analysis of texts using discourse structure. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11, pages 1061–1070, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0717-8. doi: 10.1145/2063576.2063730. URL http://doi.acm.org/10.1145/2063576.2063730.

[46] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006. ISSN 0036-8075. doi: 10.1126/science.1127647. URL http://science.sciencemag.org/content/313/5786/504.

[47] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer and Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press, 2001.

[48] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Comput., 9 (8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[49] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. CoRR, abs/1503.03244, 2015. URL http://arxiv.org/abs/1503.03244.

[50] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014073. URL http://doi.acm.org/10.1145/1014052.1014073.

[51] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management, CIKM '13, pages 2333–2338, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2263-8. doi: 10.1145/2505515.2505665. URL http://doi.acm.org/10.1145/2505515.2505665.

[52] J. D. Hunter. Matplotlib: A 2d graphics environment. Computing In Science & Engineering, 9(3):90–95, 2007.

[53] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/. [Online; accessed 2016-09-16].

[54] Daniel Jurafsky and James H. Martin. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000. ISBN 0130950696.

[55] Nobuhiro Kaji and Masaru Kitsuregawa. Building lexicon for sentiment analysis from massive collection of html documents. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 1075–1083, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D07-1115.

[56] Soo-Min Kim and Eduard Hovy. Determining the sentiment of opinions. In Proceedings of the 20th International Conference on Computational Linguistics, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1220355.1220555. URL http://dx.doi.org/10.3115/1220355.1220555.

[57] Yoon Kim. Convolutional neural networks for sentence classification. CoRR, abs/1408.5882, 2014. URL http://arxiv.org/abs/1408.5882.

[58] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

[59] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. CoRR, abs/1506.06726, 2015. URL http://arxiv.org/abs/1506.06726.

[60] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, page 2012.

[61] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, pages 2267–2273. AAAI Press, 2015. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=2886521.2886636.

[62] Thomas K Landauer. On the computational basis of learning and cognition: Arguments from {LSA}. volume 41 of Psychology of Learning and Motivation, pages 43 – 84. Academic Press, 2002. doi: http://dx.doi.org/10.1016/S0079-7421(02)80004-4. URL http://www.sciencedirect.com/science/article/pii/S0079742102800044.

[63] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. CoRR, abs/1405.4053, 2014. URL http://arxiv.org/abs/1405.4053.

[64] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units. CoRR, abs/1504.00941, 2015. URL http://arxiv.org/abs/1504.00941.

[65] S. Li, H. Zhang, W. Xu, G. Chen, and J. Guo. Exploiting combined multi-level model for document sentiment analysis. In Pattern Recognition (ICPR), 2010 20th International Conference on, pages 4141–4144, Aug 2010. doi: 10.1109/ICPR.2010.1007.

[66] Sheng-Tun Li and Fu-Ching Tsai. A fuzzy conceptualization model for text mining with application in opinion polarity classification. Knowledge-Based Systems, 39:23 – 33, 2013. ISSN 0950-7051. doi: http://dx.doi.org/10.1016/j.knosys.2012.10.005. URL http://www.sciencedirect.com/science/article/pii/S0950705112002742.

[67] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09, pages 375–384, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-512-3. doi: 10.1145/1645953.1646003. URL http://doi.acm.org/10.1145/1645953.1646003.

[68] Bing Liu. Sentiment Analysis Mining Opinions, Sentiments, and Emotions. Cambridge University Press, 2015.

[69] L. I. RozonoÃšr M. A. Aizerman, Ã. M. Braverman. Theoretical foundation of potential functions method in pattern recognition, chapter 25, pages 917–936. Avtomat. i Telemekh, 1964.

[70] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT '11, pages 142–150, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9. URL http://dl.acm.org/citation.cfm?id=2002472.2002491.

[71] Masoud Makrehchi and Mohamed S. Kamel. Automatic Extraction of Domain-Specific Stopwords from Labeled Documents, pages 222–233. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-78646-7. doi: 10.1007/978-3-540-78646-7_22. URL http://dx.doi.org/10.1007/978-3-540-78646-7_22.

[72] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press, New York, 2008.

[73] Yi Mao and Guy Lebanon. Isotonic conditional random fields and local sentiment flow. In Advances in Neural Information Processing Systems, 2007.

[74] James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization.

[75] Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. Structured models for fine-to-coarse sentiment analysis. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 432–439, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P07-1055.

[76] Merriam-Webster Online. Merriam-Webster Online Dictionary, 2009. URL http://www.merriam-webster.com.

[77] Grégoire Mesnil, Tomas Mikolov, Marc'Aurelio Ranzato, and Yoshua Bengio. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. CoRR, abs/1412.5335, 2014. URL http://arxiv.org/abs/1412.5335.

[78] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. CoRR, abs/1301.3781, 2013. URL http://arxiv.org/abs/1301.3781.

[79] Thomas M. Mitchell. Machine Learning. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN 0070428077, 9780070428072.

[80] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2924–2932. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5422-on-the-number-of-linear-regions-of-deep-neural-networks.pdf.

[81] Rodrigo Moraes, JoãO Francisco Valiati, and Wilson P. GaviãO Neto. Document-level sentiment classification: An empirical comparison between svm and ann. Expert Syst. Appl., 40(2):621–633, February 2013. ISSN 0957-4174. doi: 10.1016/j.eswa.2012.07.059. URL http://dx.doi.org/10.1016/j.eswa.2012.07.059.

[82] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In AISTATSâ05, pages 246–252, 2005.

[83] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In In Proceedings of Conference on Empirical Methods in Natural Language Processing, 2004.

[84] Vivek Narayanan, Ishan Arora, and Arjun Bhatia. Fast and accurate sentiment classification using an enhanced naive bayes model. CoRR, abs/1305.6143, 2013. URL http://arxiv.org/abs/1305.6143.

[85] Travis E. Oliphant. Guide to NumPy. CreateSpace Independent Publishing Platform, USA, 2nd edition, 2015. ISBN 151730007X, 9781517300074.

[86] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. IEEE/ACM Trans. Audio, Speech and Lang. Proc., 24(4):694–707, April 2016. ISSN 2329-9290. URL http://dl.acm.org/citation.cfm?id=2992449.2992457.

[87] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL '05, pages 115–124, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219855. URL http://dx.doi.org/10.3115/1219840.1219855.

[88] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. Found. Trends Inf. Retr., 2(1-2):1–135, January 2008. ISSN 1554-0669. doi: 10.1561/1500000011. URL http://dx.doi.org/10.1561/1500000011.

[89] Bo Pang, Lillian Lee, and Shivakumar 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, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118693.1118704. URL http://dx.doi.org/10.3115/1118693.1118704.

[90] Luca Pasa and Alessandro Sperduti. Pre-training of recurrent neural networks via linear autoencoders. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 3572–3580. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5271-pre-training-of-recurrent-neural-networks-via-linear-autoencoders.pdf.

[91] Razvan Pascanu, Yann N. Dauphin, Surya Ganguli, and Yoshua Bengio. On the saddle point problem for non-convex optimization. CoRR, abs/1405.4604, 2014. URL http://arxiv.org/abs/1405.4604.

[92] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. J. Mach. Learn. Res., 12:2825–2830, November 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2078195.

[93] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[94] Livia Polanyi and Annie Zaenen. Contextual Valence Shifters, pages 1–10. Springer Netherlands, Dordrecht, 2006. ISBN 978-1-4020-4102-0. doi: 10.1007/1-4020-4102-0_1. URL http://dx.doi.org/10.1007/1-4020-4102-0_1.

[95] Rudy Prabowo and Mike Thelwall. Sentiment analysis: A combined approach. Journal of Informetrics, 3(2):143 – 157, 2009. ISSN 1751-1577. doi: http://dx.doi.org/10.1016/j.joi.2009.01.003. URL http://www.sciencedirect.com/science/article/pii/S1751157709000108.

[96] & K. R. Scherer R. J. Davidson, H. Goldsmith. Appraisal Processes in Emotion, chapter 24, pages 24–42. Oxford University Press, 2003.

[97] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. CoRR, abs/1403.6382, 2014. URL http://arxiv.org/abs/1403.6382.

[98] Jonathon Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In Proceedings of the ACL Student Research Workshop, ACLstudent '05, pages 43–48, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1628960.1628969.

[99] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[100] Jean Y Wu Jason Chuang Christopher D Manning Andrew Y Ng Richard Socher, Alex Perelygin and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In In Proceedings of the conference on empirical methods in natural language processing (EMNLP), volume 1631, page 1642.Citeseer, 2013.

[101] Richard Socher and Eric H. Huang and Jeffrey Pennington and Andrew Y. Ng and Christopher D. Manning. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In Advances in Neural Information Processing Systems 24. 2011.

[102] Van Rijsbergen. Information retrieval. PhD thesis, dept. of computer science, university of glasgow., 1979.

[103] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL http://dl.acm.org/citation.cfm?id=104279.104293.

[104] M. Rushdi Saleh. Experiments with svm to classify opinions in different domains. Expert Systems with Applications, 38(12):14799 – 14804, 2011. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2011.05.070. URL http://www.sciencedirect.com/science/article/pii/S0957417411008542.

[105] Bahar Salehi, Paul Cook, and Timothy Baldwin. A word embedding approach to predicting the compositionality of multiword expressions. In Rada Mihalcea, Joyce Yue Chai, and Anoop Sarkar, editors, HLT-NAACL, pages 977–983. The Association for Computational Linguistics, 2015. ISBN 978-1-941643-49-5. URL http://dblp.uni-trier.de/db/conf/naacl/naacl2015.html#SalehiCB15.

[106] Bernhard Scholkopf and Alexander J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

[107] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. Trans. Sig. Proc., 45(11):2673–2681, November 1997. ISSN 1053-587X. doi: 10.1109/78.650093. URL http://dx.doi.org/10.1109/78.650093.

[108] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM '14, pages 101–110, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2598-1. doi: 10.1145/2661829.2661935. URL http://doi.acm.org/10.1145/2661829.2661935.

[109] H.T. Siegelmann and E.D. Sontag. On the computational power of neural nets. Journal of Computer and System Sciences, 50(1):132 – 150, 1995. ISSN 0022-0000. doi: http://dx.doi.org/10.1006/jcss.1995.1013. URL http://www.sciencedirect.com/science/article/pii/S0022000085710136.

[110] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11, pages 151–161, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL http://dl.acm.org/citation.cfm?id=2145432.2145450.

[111] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11, pages 151–161, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL http://dl.acm.org/citation.cfm?id=2145432.2145450.

[112] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1):1929–1958, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

[113] S. S. Stevens, J. Volkmann, and E. B. Newman. A Scale for the Measurement of the Psychological Magnitude Pitch. 8(3):185–190, January 1937. doi: 10.1121/1.1915893. URL http://dx.doi.org/10.1121/1.1915893.

[114] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[115] J.A.K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle. Weighted least squares support vector machines: robustness and sparse approximation. Neurocomputing, 48(1â4): 85 – 105, 2002. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/S0925-2312(01)00644-0. URL http://www.sciencedirect.com/science/article/pii/S0925231201006440.

[116] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. Comput. Linguist., 37(2):267–307, June 2011. ISSN 0891-2017. doi: 10.1162/COLI_a_00049. URL http://dx.doi.org/10.1162/COLI_a_00049.

[117] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. CoRR, abs/1503.00075, 2015. URL http://arxiv.org/abs/1503.00075.

[118] Luke Kien-Weng Tan, Jin-Cheon Na, Yin-Leng Theng, and Kuiyu Chang. Sentence-Level Sentiment Polarity Classification Using a Linguistic Approach, pages 77–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-24826-9. doi: 10.1007/978-3-642-24826-9_13. URL http://dx.doi.org/10.1007/978-3-642-24826-9_13.

[119] Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. Sentiment in twitter events. J. Am. Soc. Inf. Sci. Technol., 62(2):406–418, February 2011. ISSN 1532-2882. doi: 10.1002/asi.21462. URL http://dx.doi.org/10.1002/asi.21462.

[120] Peter D. Turney. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073153. URL http://dx.doi.org/10.3115/1073083.1073153.

[121] Peter D. Turney and Michael L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. ACM Trans. Inf. Syst., 21(4):315–346, October 2003. ISSN 1046-8188. doi: 10.1145/944012.944013. URL http://doi.acm.org/10.1145/944012.944013.

[122] Robertson S. Porter M. Van Rijsbergen, C. New models in probabilistic information retrieval. University of Cambridge, Computer Laboratory, 1980.

[123] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. CoRR, abs/1411.4555, 2014. URL http://arxiv.org/abs/1411.4555.

[124] Gang Wang, Jianshan Sun, Jian Ma, Kaiquan Xu, and Jibao Gu. Sentiment classification: The contribution of ensemble learning. Decis. Support Syst., 57:77–93, January 2014. ISSN 0167-9236. doi: 10.1016/j.dss.2013.08.002. URL http://dx.doi.org/10.1016/j.dss.2013.08.002.

[125] Sida Wang and Christopher 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, ACL '12, pages 90–94, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390665.2390688.

[126] M. P. Wellman and M. Henrion. Explaining 'explaining away'. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(3):287–292, Mar 1993. ISSN 0162-8828. doi: 10.1109/34.204911.

[127] Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. Neural Networks, 1(4):339 – 356, 1988. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/0893-6080(88)90007-X. URL http://www.sciencedirect.com/science/article/pii/089360808890007X.

[128] Janyce Wiebe and Ellen Riloff. Creating Subjective and Objective Sentence Classifiers from Unannotated Texts, pages 486–497. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-30586-6. doi: 10.1007/978-3-540-30586-6_53. URL http://dx.doi.org/10.1007/978-3-540-30586-6_53.

[129] Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. Learning subjective language. Comput. Linguist., 30(3):277–308, September 2004. ISSN 0891-2017. doi: 10.1162/0891201041850885. URL http://dx.doi.org/10.1162/0891201041850885.

[130] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. CoRR, abs/1511.08198, 2015. URL http://arxiv.org/abs/1511.08198.

[131] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05, pages 347–354, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220619. URL http://dx.doi.org/10.3115/1220575.1220619.

[132] Bishan Yang and Claire Cardie. Joint modeling of opinion expression extraction and attribute classification. Transactions of the Association for Computational Linguistics, 2:505–516, 2014. ISSN 2307-387X. URL https://transacl.org/ojs/index.php/tacl/article/view/409.

[133] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. CoRR, abs/1409.2329, 2014. URL http://arxiv.org/abs/1409.2329.

[134] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. CoRR, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

[135] Lei Zhang and Bing Liu. Identifying noun product features that imply opinions. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2, HLT '11, pages 575–580, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL http://dl.acm.org/citation.cfm?id=2002736.2002849.

[136] Ziqiong Zhang, Qiang Ye, Zili Zhang, and Yijun Li. Sentiment classification of internet restaurant reviews written in cantonese. Expert Systems with Applications, 38(6):7674 – 7682, 2011. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2010.12.147. URL http://www.sciencedirect.com/science/article/pii/S0957417410015101.

[137] J. Zhao, M. Lan, Z. Y. Niu, and Yue Lu. Integrating word embeddings and traditional nlp features to measure textual entailment and semantic relatedness of sentence pairs. In 2015 International Joint Conference on Neural Networks (IJCNN), pages 1–7, July 2015. doi: 10.1109/IJCNN.2015.7280462.