UNIVERSITÀ DEGLI STUDI DI MILANO

DOCTORAL SCHOOL OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE
(XXVIII CYCLE)

A DEEP LEARNING APPROACH FOR SENTIMENT ANALYSIS

CANDIDATE
Dott. Michele Di Capua

ADVISOR Prof. Alfredo Petrosino

DIRECTOR OF DOCTORAL SCHOOL Prof. Paolo Boldi

ACADEMIC YEAR 2016/2017

dedicated to my Dad

Acknowledgements

I really would like to express my deep gratitude to my advisor Prof. Alfredo Petrosino for his patient guidance during these years, for his enthusiastic encouragement and useful critiques of this research work.

I also wish to thank various people for their contribution to this work, as my reviewers Prof. Michele Ceccarelli, Prof. Alessandro Vinciarelli and Prof. Witold Pedrycz, for their valuable and constructive suggestions.

A special thanks to Dr. Emanuel Di Nardo for his technical help.

Finally, I'd like to thank also my colleagues Ihsan Ullah and Srdjan Matic for all the conversations, opinions and also practical hints exchanged during this experience.

Abstract

Sentiment Analysis refers to the process of computationally identifying and categorizing opinions expressed in a piece of text, in order to determine whether the writer's attitude towards a particular topic or product is positive, negative, or even neutral.

The views expressed and its related concepts, such as feelings, judgments, and emotions have become recently a subject of study and research in both academic and industrial areas. In particular the beginning of these studies and their rapid growth coincide precisely with the advent and popularity of social networks and social media, in which information such opinions, reviews, discussions and comments proliferate.

The spread of social networks has still created also significant risks especially among young teenagers who are also the main users of the web.

One of these risks, that is now growing strongly, is **cyber bullying**. More exactly the term cyber bullying indicates acts of harassment, humiliation, and general indirect aggressive actions, carried out by e-mail, instant messaging, but more commonly through posting comments activity on social networks. Our assumption here is that cyber bullying detection can be treated as a particular case of Sentiment Analysis task.

Unfortunately language comprehension of user comments, especially in social networks, is inherently complex to computers. The ways in which humans express themselves with natural language are nearly unlimited and informal texts is riddled with typos, misspellings, badly set up syntactic constructions and also specific symbols (e.g. hashtags in Twitter) which exponentially

complicate this task. Furthermore, humans could learn new words by the context in which they appear but for computers extracting this information and using it appropriately is not really easy.

Many of the studies in literature have adopted machine learning approaches to solve Sentiment Analysis tasks. Since the performance of machine learning algorithms heavily depends on the choices of data representation, many recent studies have been focused on the creation of automatic feature extractors due to the huge effort in building hand-crafted features.

Recently, **deep learning** approaches are emerging as powerful computational models that discover intricate semantic representations of texts automatically from data without hand-made feature engineering. These approaches have improved the state-of-the-art in many Sentiment Analysis tasks including sentiment classification of sentences or documents, sentiment lexicon learning and also in more complex problems as cyber bullying.

The contributions of this work are twofold. First, related to the general Sentiment Analysis problem we propose a semi-supervised neural network model, based on **Deep Belief Networks**, able to deal with data uncertainty for text sentences in Italian language. We test this model against some datasets from literature related to movie reviews, adopting a vectorized representation of text and exploiting methods from **Natural Language Processing** (NLP) pre-processing.

Second, assuming that the cyber bullying phenomenon can be treated as a particular Sentiment Analysis problem, we propose an unsupervised approach to automatic cyber bullying detection in social networks, based both on **Growing Hierarchical Self Organizing Map** (GH-

SOM) and on a specific features model, showing that our solution can achieve interesting results, respect to classical supervised approaches.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"If you can't explain it simply,
you don't understand it well enough."

- Albert Einstein

This chapter provides a general introduction to the thesis objectives and a short outline of the structure of this work, with a brief description of the contents of each chapter.

1.1 Problem definition and challenges

Sentiment analysis attempts to identify and analyze opinions and emotions from sentences providing a classification of text.

The ability to analyze and measure the "feeling" of users respect to a generic product or service, expressed in textual comments, can be an interesting element of evaluation that can guide business dynamics, both in the industry area and also in marketing. Automatic text classification of texts into pre-defined categories has witnessed an increasing interest in the last years, due to the huge availability of documents in digital form.

Some examples of sentences that express a feeling or an opinion are: *"This movie is really ugly and the direction is very bad"*, or *"The story is pretty but the director could do more"*.

In the research community the recent dominant approach to general text classification and Sentiment Analysis is increasingly based on machine learning techniques: a general inductive process that automatically builds a classifier by learning the main features of the categories from a set of pre-classified documents. The advantages of this approach over the knowledge engineering classical approach (consisting in the manual definition of a classifier by domain experts) are a very good effectiveness, a considerable savings in terms of expert manpower, but also a straightforward portability to different domains. Depending on the problem statement, in Sentiment Analysis tasks, classifiers like SVM (Support Vector Machine) or Naive Bayes, have shown good performance accuracy, provided proper feature engineering and also dedicated pre-processing steps, to be executed before the classification process.

Also, these traditional approaches are lacking in face of structural and cultural subtleties in the written language. For instance, negating a highly positive phrase can completely reverse its sentiment, but unless we can efficiently present the structure of the sentence in the feature set, we will not be able to capture this effect. On a more abstract level, it will be quite challenging for a machine to understand sarcasm in a review. The classic approaches to sentiment analysis and natural language processing are heavily based on engineered features, but it is very difficult to hand-craft features to extract properties mentioned above. But also, due to the dynamic nature of the language, those features might become obsolete in a very short amount of time. We need a different approach to overcame these problems.

In 2006, G. Hinton introduces the Deep Belief Networks (DBNs) using a learning algorithm that in a greedy way performs a training of the network one layer at a time. Hinton proposes to use for each level of the deep network a particular neural network, called Restricted Boltzmann Machines (RBMs), which is trained by using an unsupervised algorithm. The results of DBNs seem to be promising, and have generated new interests in this field of Machine Learning.

Recently, deep learning algorithms have shown impressive performance in natural language processing applications including sentiment analysis across multiple data sets. These models do not need to be provided with predefined features hand-picked by an engineer, but they can learn sophisticated features from the data set by themselves. Although each single unit in these neural networks is fairly simple, by stacking layers of non-linear units at the back of each other, these models are capable of learning highly sophisticated decision boundaries. Words are represented in a high dimensional vector space, and the feature extraction is left to the neural network [4]. As a result, these models can map words with similar semantic and syntactic properties to nearby locations in their coordinate system, in a way which is reminiscent of understanding the meaning of words. Architectures like Recursive Neural Networks are also capable of efficiently understanding the structure of the sentences [5].These characteristics make deep learning models a natural fit for a task like sentiment analysis.

1.2 Contributions of this thesis

The contributions made by this work are different.

We started exploring machine learning approaches in Sentiment Analysis tasks, providing a taxonomy of all the available techniques adopted. Then, we proposed an unsupervised network model based on Deep Belief Networks in order to classify movie reviews data sets, taken from literature, adopting a vectorized representation of text and methods from Natural Language Processing (NLP).

Due to the lack of resources in other language than English we also provide two new data sets in Italian language, built from two different public web sources.

As a particular case of Sentiment Analysis task we investigate also the cyber bullying problem on social networks, and we propose a new feature set model to be used in an unsupervised approach, with Growing Hierarchical Self Organizing Map (GHSOM).

Finally, we compare our models with different models and data sets taken from literature in order to evaluate the efficiency of our proposals..

1.3 Thesis outline

The thesis is organized in seven chapters, three appendices, and a bibliography.

Chapter 1: In the first chapter we introduce the main problem, the related challenges and we provide an outline of the thesis.

Chapter 2: The second chapter introduces the Sentiment Analysis task, with its definitions and terms, and a state of the art of the applied techniques in this field.

We discuss this topic mainly from a Machine Learning perspective, focusing also on the current limitations of Sentiment Analysis classification tasks.

Chapter 3: In the third chapter we describe our proposed semi-supervised model, its motivations, algorithms and a general architecture. We discuss the advantages of this approach in discover automatically hidden semantic structure over data, with particular reference to the Sentiment Analysis task and to the proposed model of deep network.

Chapter 4: In the fourth chapter we discuss the cyber bullying social phenomenon, giving its definition and issues. About that, we propose a new unsupervised model, based on hand crafted engineered features and a Growing Hierarchical Self Organizing Map, in order to detect automatically bully traces in social networks.

Chapter 5: In the fifth chapter we briefly introduce Natural Language Processing techniques, with a particular focus on the subset of methods adopted in this thesis in the pre-processing stage of the textual data analysis.

Chapter 6: In the sixth chapter we describe our datasets, our experiments, and we show the results obtained with our models, for both the general Sentiment Analysis task and for the automatic detection of cyber bullying traces.

Chapter 7: In this last chapter we report our final remarks on the thesis objectives and propose also some future directions to extend this work.

Appendix A: At the end of this thesis, in the appendix A section, we also define and propose a distributed software architecture for Sentiment Analysis activity, that can be used to practically deal with big data commonly produced by social networks.

Appendix B: The appendix B reports a list of produced publications related to the arguments of this thesis.

Appendix C: The appendix C contains a brief description of the software tools used in this work .

Chapter 2

Sentiment Analysis

"It has become appallingly obvious that our technology has exceeded our humanity."

- Albert Einstein

In this chapter we recall the definition of Sentiment Analysis, its related terms and concepts, and a short review of commons approaches in this field, based on both Machine Learning and Lexicon methods. We also discuss current limitations of this task and its complexities, and what could be a possible solution to mitigate these complexities, as the proposed model in this thesis.

2.1 Terms and definitions

In order to obtain a formal definition of Sentiment Analysis let's first consider the definition of the single word *sentiment*. From the Cambridge vocabulary we can read that a sentiment is defined as *"a thought, opinion, or idea based on a feeling about a situation, or a way of thinking about something"*. It is clear from this definition that a sentiment is strictly related to a personal or subjective (not factual or objective) opinion. Wiebe in

[6] more formally defines the subjectivity of a sentence as *"the set of all the elements that describe the emotional state of the author"*. Typical subjectivity clues can be considered: assumptions, beliefs, thoughts, experiences, and also opinions. In summary, we can say that a *feeling* or *sentiment*, related to a given text, can be defined as the set of subjective expressions. These expression can commonly be measured in terms of positive, neutral or negative orientation.

With this premise Sentiment Analysis, introduced first in 2003 in [7], describes the process of evaluating the *polarity* expressed by a set of documents in an automatic way. The term polarity here still refers to the orientation of the author expressed in the sentence.

Definitions and terms in this field have been subsequently refined. In [8], Bing Liu defines formally an opinion as a binomial expression made of two fundamental parts:

- a **target g**, also called topic, which represent an entity (or one of its aspects);
- a **sentiment s**, expressed on the target, which can assume positive, negative or even neutral values. Normally the sentiment can be expressed also using a number (score), or by a kind of ranking (e.g. from 1 to 5 stars). Positive or negative terms represent also the so called polarity of the expressed sentiment;

Considering the following product review, made by different sentences, related to a camera:

1. *"I bought this camera six months ago."*
2. *"It's really wonderful."*
3. *"The picture quality is really amazing."*
4. *"The battery life is good."*

5. *"However, my wife thinks it's too heavy for her."*

Looking at these sentences, that belong to the same review, it's possible to note some basic features. The review contains both positive and negative sentences on the same product. The phrase number 2 expresses a very positive opinion on the whole product. The phrase number 3 expresses still a very positive opinion, but only on one aspect of the product. The phrase number 4 expresses a good opinion but always on a single aspect of the product. The phrase number 5 expresses instead a negative opinion on a precise aspect (the weight) of the product.

We can now formally extends the definition of opinion as a quadruple:

$$Op = (g, s, h, t) \tag{2.1}$$

where g is the opinion, s is the sentiment, h is the opinion holder and t is a time stamp related to the time at which the opinion has been expressed.

It's worth to note that in this field of studies, the terms *opinion* and *opinion mining* [9] are often used in a interchangeable way, instead of *sentiment* or *sentiment analysis*. Even if the terms are similar the meaning of the underneath processes are different. The process related to automatic analysis of opinions, also called **Opinion Mining**, is more focused in finding both a list of aspects of a certain product (quality, features, etc.) but also in extracting opinions on each of these single aspects (poor, good, excellent).

To ensure that the text classification task could be really feasible and significant when applied, existent literature usually adopts an important assumption, sometimes implicitly, that has been defined by Liu et al. in

[8]: in the classification activity of a document *d* it is assumed that this document expresses an opinion (e.g., a product review) only on a single subject *s* or entity, and this opinion is expressed by a single author *h*. In practice, if a document contains opinions on different entities, then these opinions can potentially be different, i.e. positive on certain entities and negatives on other entities. Therefore, it makes no sense to assign a unique feeling or sentiment to the entire document. This assumption is still valid in the case of product or service reviews, that are generally wrote by a single author and are usually related to a single target.

While it's possible to approach Sentiment Analysis in a few ways, commonly in literature we have 3 typical levels of analysis:

- **Document Level**: at this level we analyze the overall sentiment expressed in the text. This level of analysis is based on the assumption that the whole document discusses only one topic and thus cannot be applied to documents that contain opinions on more than one entity.
- **Sentence Level**: at this detail we examine the sentiment expressed in single sentences. This kind of analysis determines whether each sentence contains a positive or negative opinion. This level of analysis can be also considered as a subjectivity classification [6], which aims to distinguish objective sentences, that simply express factual information, from subjective sentences, that express personal views and opinions. However, it's worth to note that subjectivity is not equivalent to sentiment as many objective sentences can imply opinions.
- **Entity and Aspect Level**: this much more gran-

ular analysis takes into consideration each opinion expressed in the content and it is generally based on the idea that an opinion consists of both a sentiment (positive or negative) and a target of opinion.

In this thesis we'll deal with classical Sentiment Analysis tasks, adopting binary polarities (positive, negative) of text, and analyzing text at a sentence level detail.

2.2 Related works

A more general Sentiment Analysis research activity, dealing with interpretation of metaphors, sentiment adjectives, subjectivity, view points, and affects, mainly started from early 2000, with some earlier works done by Hatzivassiloglou et al. [10]; Hearst [11], Wiebe[12][13], Bruce [14].

Specific works that contain the term Sentiment Analysis or Opinion Mining appeared in the same years, with Turney [15], Das et al. [16], Morinaga et al. [17], Pang et al. [18], Tong [19], Wiebe [20].

The work done by Turney [15], on review classification, presents an approach based on the distance measure of adjectives found in text, using preselected words with known polarity as *excellent* and *poor*. The author presents an algorithm based on three steps which processes documents without human supervision. First, the adjectives are extracted along with a word that provides contextual information. Words to extract are identified by applying predefined patterns (for instance: adjective-noun or adverb-noun etc.). Next, the semantic orientation is measured. This is done by measuring the distance from words of known polarity. The mutual dependence between two words is found by analysis of hit count with the AltaVista search engine for documents that contain two words in a certain proximity of each other. At the end the algorithm counts the average semantic orientation for all word pairs and classifies a review as recommended or not.

In contrast, Pang et al. [18] present a work based on classic topic classification techniques. The proposed approach aims to test whether a selected group of machine learning algorithms can produce good result when Sentiment Analysis is perceived as document topic analysis with two topics: positive and negative. Authors present results for experiments with: Naive Bayes, Maximum Entropy and Support Vector Machine algorithms that will be discussed in the next paragraph. Interestingly the performed tests have shown results comparable to other solutions ranging from 71% to 85% depending on the method and test data sets.

Riloff and Wiebe [21] put most of impact in their work on the task of subjective sentences identification. They propose a method that at bootstrap uses a high precision (and low recall) classifiers to extract a number of subjective sentences. During this phase sentences are labeled by two classifiers: first for high confidence subjective sentences, second for high confidence objective sentences. The sentences that are not clearly classified into any category are left unlabeled and omitted at this stage. Both of the classifiers are based on preset list of words that indicate sentence subjectivity. The subjective classifier looks for the presence of words from the list, while the objective classifier tries to locate sentences without those words. According to the results presented by authors their classifiers achieve around 90% accuracy during the tests. In the second step, the gathered data is used a for training an extraction algorithm that generates patterns for subjective sentences. The patterns are used to extract more sentences in the same text. The presented method has such split in order to increase recall after the initial bootstrap phase (however, as expected, author report the precision to fall between 70-80%).

In opposition to it, the work done by Yu and Hatzivassiloglou [22] discusses both sentence classification (subjective/objective) and orientation (positive / negative / neutral). For the the first step of sentence classification, authors present test results for three different algorithms:sentence similarity detection, Naive Bayes classification and Multiple Naive Bayes classification. In the second step of sentence orientation recognition authors use a technique similar to the one used by Turney [15] for document level sentiment analysis. The main difference is that the algorithm is extended to use more then two (excellent/poor) base words to which all others are compared.

Sometimes authors go even further and present methods for specific text format, for instance reviews where positive and negative features are explicitly separated is different areas.

A similar approach is presented by Hu and Liu in their work about customer reviews analysis [23]. In their research authors present opinion mining based on feature frequency. Only the most frequent features, recognized by precessing many review, are taken into consideration during summary generation.

During last years a very active research group in Sentiment Analysis has been the Stanford NLP group. From this group several reference papers has been produced. In [24], Socher et al. proposed a semi-supervised approach based on recursive autoencoders for predicting sentiment distributions. The method learns vector space representation for multi-word phrases and exploits the recursive nature of sentences. In [25], Socher et al. proposed a matrix-vector recursive neural network model for semantic compositionality, which has the ability to learn compositional vector representations for phrases and sentences of arbitrary length. The vector captures the inherent meaning of the constituent, while the matrix captures how the meaning of neighboring words and phrases are changed. Then in [26] the same authors propose the Recursive Neural Tensor Network (RNTN) architecture, which represents a phrase through word vectors and a parse tree and then compute vectors for higher nodes in the tree using the same tensor-based composition function.

Regarding convolutional networks for NLP tasks, Collobert et al. [27] use a convolutional network for the semantic role labeling task with the goal avoiding excessive task-specific feature engineering. In [28], the same author use a similar network architecture for syntactic parsing.

2.3 A taxonomy of possible approaches

The Sentiment Analysis topic has become, during the last years, of great interest for scientific community. It become known that a considerable boost, in this scope, has gave from the huge amount of available data by the increasing usage of social platforms for information sharing and exchange. Consequently many scientific articles have been produced on this topic and today it results still increasing. It become useful make a short examination of main approaches proposed and developed, during the years, in this area. In figure 2.1 is shown a taxonomy of principal approaches adopted in the field of Sentiment Analysis.

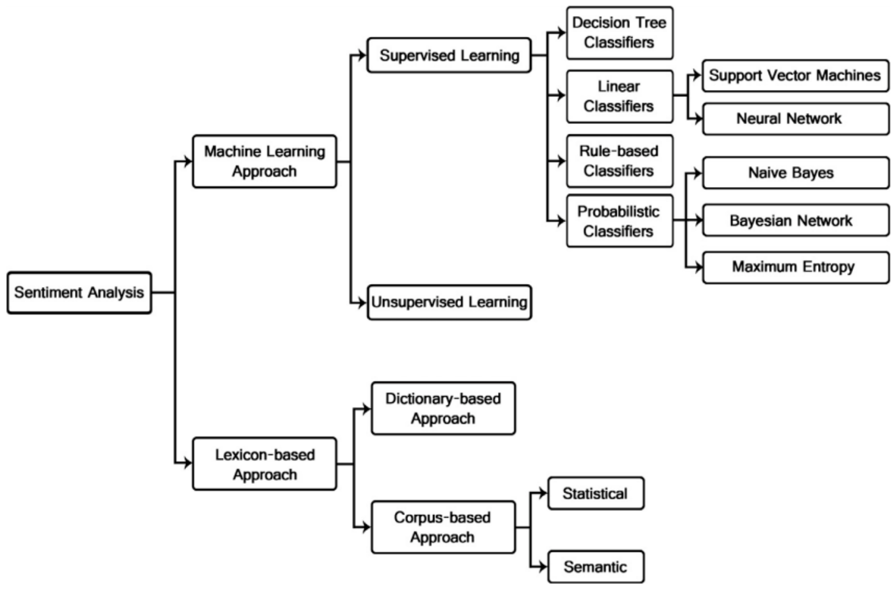


Figure 2.1: Taxonomy of principal approaches adopted in the field of Sentiment Analysis.

Below in this chapter we will provide a summary of methods used in literature, with greater attention to *feature selection* and *sentiment classification* techniques. Approaches in figure 2.1 can be divided, initially, in two macro-category, the former based on Machine Learning, and the latter on Lexicon.

2.3.1 Features selection techniques

Selecting relevant features and deciding how to encode them for a learning method can have an enormous impact on the learning method's ability to extract a good model. Much of the interesting work in building a classifier is deciding what features might be relevant, and how we can represent them. Although it's often possible to get decent performance by using a fairly simple and obvious set of features, there are usually significant gains to be had by using carefully constructed features based on a thorough understanding of the task at hand. However, there are usually limits to the number of features that you should use with a given learning algorithm if you provide too many features, then the algorithm will have a higher chance of relying on idiosyncrasies of your training data that don't generalize well to new examples. This problem is known as over-fitting, and can be especially problematic when working with small training sets.

In works [18] and [29] has been proposed a supervised approach to select features, starting from comparing different machine learning techniques (Naive Bayes, Maximum Entropy, SVM) and considering different kind of features from input models, such as unigrams, bigrams and combination of these, usage of POS, term position and exclusively adjectives. Dataset used by Pang Lee et al. is a list of film reviews, with two classes of "polarity" assigned to documents: positive or negative. Results of these comparisons are resumed in the table 2.1, that represents an important reference for the literature. From these shown results it is possible to derive some helpful indications that issue from experiments such as:

a. Presence of a feature is more important of its frequency.
b. Usage of bigrams or trigrams doesn't seems to improve accuracy.
c. Accuracy is improved if are considered all terms frequency belongs to any tag, rather than considers only adjectives.

Features	**Number of features**	**Frequency or Presence?**	**NB**	**ME**	**SVM**
Unigrams	16165	Freq.	**78.7**	N/A	72.8
Unigrams	16165	Pres.	81.0	80.4	**82.9**
Unigrams+bigrams	32330	Pres.	80.6	80.8	**82.7**
Bigrams	16165	Freq.	77.3	**77.4**	77.1
Unigrams+POS	16695	Freq.	81.5	80.4	**81.9**
Adjectives	2633	Freq.	77.0	**77.7**	75.1
Top 2633 unigrams	2633	Freq.	80.3	81.0	**81.4**
Unigrams+position	22430	Freq.	81.0	80.0	**81.6**

Table 2.1: Accuracy comparison using different classifier on film reviews dataset.

In the research activity conducted during last years different learning experiments were developed and, also, different kind of features were engineered and applied. As for others supervised learning applications, also in our case the key for an efficient opinion classification is the engineering of a representative set of features. Some of most commons used in supervised models (specifically adopted for text classification), referenced in literature, are:

- **Terms and their frequency.** These features are based or on single words (unigrams) or on N-grams (more words) associated, also, with frequency appearance, both in the whole corpus and on single document. These features are, de-facto, those more frequently adopted in text classification problems. In some cases, also the position of the word in the sentence can be as much useful. Technique based on TF-IDF[1] can, for example, be applied when the model is also based on the term frequency analysis. It has been demonstrated that usage of this function

[1]Weight function **tf-idf** (*term frequency-inverse document frequency*) is a function used in Information Retrieval to measure the importance of a term respect to a document or a collections of documents. This function increase proportionally to the number of times the term is contained in the document, but increase inversely proportional with frequency of the term in the collection. The base idea of this function is to give more importance to terms that appear in the document, but also have a lower frequency (Wikipedia).

can carry some benefits also in task of Sentiment Analysis.

- **POS (Part Of Speech).** Components and parts, of a speech of each word, result to be also really important in Sentiment Analysis task. Equal words, but which play a different role in the text can be treated and taken into account in a different manner. For example, it is demonstrated that adjectives are fundamentals indicators of expressed opinions. Therefore some researchers have treated adjectives as special features. We can observe in table 2.2 the mapping scheme of tag POS in italian language, which have been adopted in this project. Some specific syntactic patterns have been considered useful both to identify the presence of adjectives and adverbs, that are possible indicators of opinion and "sentiment", and to identify potential negations or so called "sentiment shifters" that potentially can "invert" the polarity in a sentence.
- **Sentence and opinion words.** Words that express an opinion are those, which in a language are used to express directly positive and negative sentiments. For example, in italian, adjective such as *buono* (good), *meraviglioso* (wonderful), *fantastico* (fantastic), are used to express positivity, instead words such as *male* (bad), *povero* (poor), *terribile* (terrible) express negativity. Although all parts of a document could be important, it is demonstrated that people use adjectives to express their sentiments and opinions. Models based only on adjectives for features creation have achieved on average good results. In [18] have achieved an accuracy of 82.8% classifying film reviews texts, using a model based

only on adjectives utilization. Adverbs taken individually usually don't express sentiments, but if used together with adjectives, they can assume a fundamental role to determinate sentence polarity. A work on this theme is developed in [30]. As well as adjectives and adverbs, it is possible to use nouns (for example *trash*, *rubbish*) or verbs (*love*, *hate*) to express at same manner positive and negative sentiments. Further single words often also idioms (sayings or particular writing style) can be used to express an opinion (for example "it was better if that actor didn't open his mouth").

- **Polarity Shifters** There are expressions used to completely change sentence polarity, from positive to negative or vice-versa. Negations are the most important class of modifiers. For example the sentence "*I don't like this camera*" is negative, also there is the term "*like*" that alone is a positive marker. This kind of construct have to be well treated with great attention because not all the time negations are used to change opinion. Handle negations represents a limitation of Sentiment Analysis, derived directly from the complexity of written languages language. In the presented work there are considered and managed some simple case of "sentiment shifting", to mitigate the negations problem. Some example of researched patterns (in form of POS Tag sequences) handled to identify a negation have been: *BNVip3B, BNB, BNRiAs, BNVip3Ss, BNPCVis3.*

TAG	**Meaning**
B	adverb
BN	negation adverb
CC	coordinate conjunction
CS	subordinate conjunction
D	determiner
DE	exclamative determiner
DI	indefinite determiner
DQ	interrogative determiner
DR	relative determiner
E	preposition
EA	articulated preposition
FB	balanced punctuation () [] ””
FC	clause boundary punctuation (:;)
FF	comma (,)
FS	sentence boundary punctuation (.?!)
I	interjection
N	cardinal number
PC	clitic pronoun
PD	demonstrative pronoun
PE	personal pronoun
PI	indefinite pronoun
PP	possessive pronoun
PQ	interrogative pronoun
PR	Pronome relativo
RI	indeterminative article
T	predeterminer
SA	abbreviation
SP	proper noun
XH	Twitter hashtag (#nlp)
XM	Twitter mentions (@obama)
XE	emoticon (smiley :-))
XX	other (equation, non classified, etc...)
S[spn]	common noun*
A[spn]	adjective*
AP[spn]	possessive adjective*
NO[spn]	ordinal number*
SW[spn]	foreign noun*
V[icdgpfm][pisf][3]	verb[mode][tense][3rd person]
VA[icdgpfm][pisf][3]	auxiliary verb[mode][tense][3rd person]
VM[icdgpfm][pisf][3]	modal verb[mode][tense][3rd person]
	*[s = singular, p = plural, n = not specified]

Table 2.2: POS Tag scheme.

2.3.2 Lexicon based approaches

Many text classification techniques are based on sets of words that directly express an opinion or a polarity. There are also phrases and idioms which in turn implicitly express opinions within a particular text. These phrases and idioms constitutes a so-called opinion lexicon. From literature it appears that three main methods used to identify and build a set of words that express an opinion: the first is a manual method, in which an expert must manually label each word. This method is usually accompanied by some automated techniques that allow the creation of hybrid approaches that will be described in this paragraph.

2.3.2.1 Dictionary-based

This approach is based on the activity of building a small set of word (organized typically in a hierarchy structure) with an associated polarity value. This set can be increased using data provided from web services such as WordNet[2], to search synonyms and antonyms of words contained in the first base set. Some possible limitations of this approach are established from the fact that the context in which words are used is not taken into account. Moreover it is also possible to find words (or synonyms) without polarity.

2.3.2.2 Corpus-based

This approach is able to solve previous problem related to find words that express opinions inside a specific context.

[2]WordNet is an open source database of semantic-lexical type for english language made by the linguist George Armitage Miller from Princeton University, that propose to organize, define and describe the concept expressed by terms. Organization of lexicon use grouping of terms with similar meaning, called "synset" (from contraction of "synonym set"), and from the linkage of their significance through different kind of relationship clearly defined. In the synset significance difference are numbered and defined. Lexicon is freely accessible online.

This method is based on models and syntactic recurrent patterns, that are used inside corpus of big dimensions to to list words expressing a certain polarity. For example is intuitive to imagine two adjectives joined by the conjunction "e (and)" express same polarity. Therefore knowing the polarity of an adjective it is possible to search new adjective (and attribute them the same polarity), if that new adjective complies with the simple syntactic pattern to be in conjunction "e" with the known adjective.

2.3.2.3 Semantic-based

Semantic approach defines polarity based on the proximity of two words, it assign same polarity to words that are semantically "near". Set of terms contained in WordNet just provides some semantic relationship among words and can be used as reference to find this "proximity" relationship. The first step of the process is to count the number of positive and negative synonym of a single word and label that word using the polarity of the most high counter. This kind of approach is used in many application to build a semantic model used for example to describe verbs, nouns and adjectives, that will be used for future text classification.

2.3.3 Machine Learning approaches

The text classification methods, based on Machine Learning, can be roughly divided into two groups: the so-called supervised methods and the unsupervised methods. Supervised methods adopt extensive use of data and documents in general, that have been previously labeled, commonly by hand. Instead, the unsupervised methods are used in those cases where it is difficult to obtain already classified documents. In case of supervised methods it is possible to find in the literature several types of proposed classifiers. In this paragraph we will give a brief review of the most adopted supervised classifiers.

2.3.3.1 Naive Bayes classifier

The Naive[3] Bayes classifier is one of the most used and simplest classifier adopted in literature, able to estimate a posterior probability of a class, given the distribution of words in a document. More specifically this model works with an approach based on an organization of data in BOW (Bag of Words) where the position of a word or the grammar isn't taken into account. The classifier used the Bayes theorem to predict probability that a set of features belongs to a specific class (label). That probability is defined as:

$$P(label|features) = \frac{P(label) * P(features|label)}{P(features)} \quad (2.2)$$

where $P(label)$ is the probability of a single label, or also, the probability that to a set of features is assigned to a label in a random manner. $P(features|label)$ is the prior probability that a set of features is assigned a given label. $P(features)$ is the prior probability of a given set of features. Assumption of Naive Bayes classifier consists in the fact that all attributes, describing a certain instance, are conditional independents among them given the category on which it belongs - translated in our case of textual analysis, the words that compose a comment are independent of among them. In reality, however, words occurring in a text are strongly related.

[3]Naive means that the classifier assumes that the document and its words are independent of each other.

For example, considering the following sentence (italian and english version) :

> *Il tuo lavoro non é per niente male.* (ITA)
> *Your work is not bad.* (ENG)

This sentence expresses a clear positive message, however, it is composed by indicators that are generally negatives (*non/not*, *male/bad*) and for which Naive Bayes classifier could interpret as markers of a negative message. The absence of a more semantic approach here represent one of the main limitation of NB classifier in Sentiment Analysis field.

2.3.3.2 Maximum Entropy classifier

The Maximum Entropy classifier is a discriminative classifier widely used in *Natural Language Processing* field and particularly adopted to solve problems of text classification, such as language and topic detection and Sentiment Analysis. The Max Entropy algorithm is based on the principle of *Maximum Entropy*: it selects the model that represent the maximum entropy (according to Shannon[4]) on the training set, among all tested models. Recalling the Bayes theorem, the Max Entropy classifier is used when there aren't information on a prior distribution of data and it is not possible to make assumptions on that distribution. Unlike Naive Bayes classifier, the Max Entropy classifier doesn't use the hypothesis of independent variables, that is the true nature of natural language where variables are words which aren't inde-

[4]Taking a source of messages X, the amount of information carried by the message increases with the increasing of its uncertainty. The greater is the knowledge of the produced message by the source, the smaller are the uncertainty, the entropy and the carried information.

pendent because they are fasten by grammatical rules; furthermore in contrast with Naive Bayes model, this approach requests an expensive training time but produces more reliable results.

Consider an example that clarifies better the principle on which this classifier is built. Suppose we want to determine the grammatical form of the italian word "*amare*" (to love). This word can assume the following forms:

- **Adjective:** *"Queste cioccolate sono tutte amare."*
- **Noun:** *"Amare é un bisogno innato dell'uomo."*
- **Verb:** *' 'I narcisisti riescono ad amare solo se stessi."*

We collect a number of samples large enough from which extract information to set the decision model. Model p we are going to build assign to word "*amare*" a probability to assume a particular grammatical significance. Without further information on data, it is possible to impose for our model p:

$$p(adjective) + p(noun) + p(verb) = 1 \tag{2.3}$$

There are countless models p for which the previous identity is valid, among them:

- $p(adjective) = 1, p(noun) = p(verb) = 0;$
- $p(noun) = 1, p(adjective) = p(verb) = 0;$
- $p(verb) = 1, p(adjective) = p(noun) = 0;$
- $p(adjective) = p(noun) = p(verb) = \frac{1}{3};$

Analyzing dataset further, we suppose to hold new information, such as each time the word "*amare*" is preceded from word "*sono*" is an adjective. This fact, added to the normalization condition, change the probabilities, reducing the number of models. The MaxEnt algorithm aim is to determine the model p most uniformly as possible (maximizing entropy), following only the information derived by data, without any further hypothesis.

2.3.3.3 Support Vector Machine

The **SVM** (**Support Vector Machine**) classifier is based on the principle of finding some linear separators within a search space, which can better divide points (data), that belongs to different classes. In figure 2.2 two different classes are shown (identified by a cross and a circle) and three hyperplanes[5] A, B, C. In the example the hyper-plane A provides the best separation among the classes, because the *normal distance* (pointed by the arrow) is the maximum possible and, for this reason, it represents the maximum margin of separation.

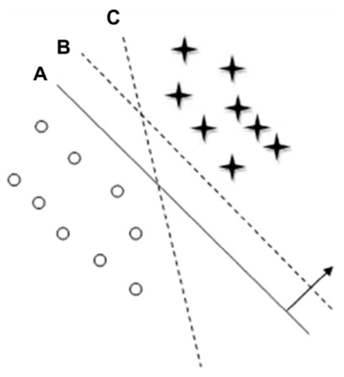


Figure 2.2: Example of linear separation among three classes by SVM.

Textual data, as in our case, are ideal data to classify with SVM, because the "*sparse*" nature of a text, in which only some features are not useful, but tend to be correlated and, generally, results organized in linearly

[5]Essentially it represents a sub-space of dimension lesser than one (n-1) respect to the space where it is contained (n). If the space has a size of 2 than its hyper-plans are two.

separable category. SVM classifier use a *non-linear decision surface*, in the original features space, building a mapping between the initial non-linear separable data and a new space with many dimensions, where features appear separable by an hyper-plane, as shown in figure 2.3

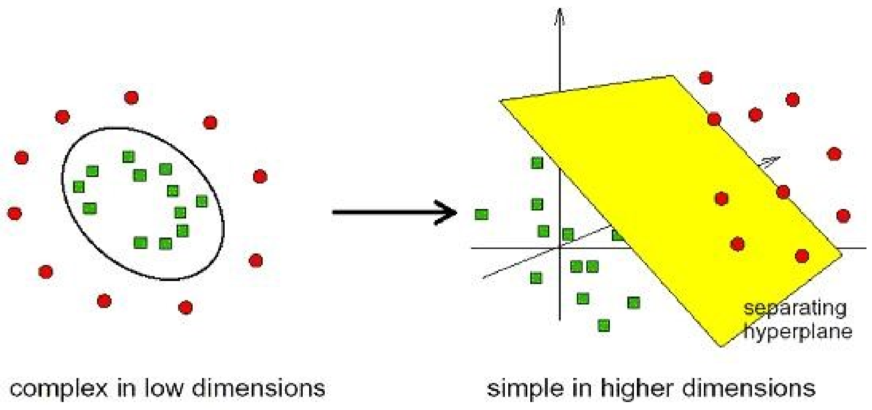


Figure 2.3: Projection of non-linear separable features from a low dimension to a higher dimension.

2.3.3.4 Artificial Neural Networks

Among supervised approaches there are also classification methods based on artificial neural networks (ANN). A neural network consists of different computational units called neurons. The input for neurons is, in the case of text analysis, typically a vector with frequency of words in each document $X_i \in X$, where X is a set of documents. There are some weights W for each neuron. The linear separation function is represented as $p_i = W * X_i$. There are also, neural networks with multiple layers for non-linear classification problems. In this case step of training become more complex because classification error must be propagated backward on different layers.

Some researchers compared SVM with some NN and they have demonstrated that neural networks achieve a better accuracy, mostly when data aren't well balanced. Furthermore neural networks can be used also with non-supervised approaches without labeled data.

In this work the proposed solution for Sentiment Analysis will be based proper on neural networks with both a an unsupervised and a semi-supervised model, that is models that extract automatically features from a set of data and then uses only a subset of labeled data to improve classification performance. Better details on this aspect will be treated in chapter 3 where will be introduced also the concepts of recent models based on "deep learning".

2.4 Language dependent analysis

The majority of current Sentiment Analysis systems unfortunately address a single language, usually English and so a related problem is a significant lack of resources as data sets [31]. An interesting survey on the state of the art on multilingual Sentiment Analysis is presented in [32]. However, with the growth of the Internet around the world, users write comments in different languages. Sentiment analysis in only single language increases the risks of missing essential information in texts written in other languages. In order to analyse data in different languages, multilingual Sentiment Analysis techniques have been developed [33]. With this, sentiment analysis frameworks and tools for different languages are being built. Thus, sentiment analysis in multiple languages is often addressed by transferring knowledge from resource-rich to resource-poor languages, because there are no resources available in other languages. The majority of multilingual sentiment analysis systems employ English lexical resources such as SentiWordNet.

Another approach is to use a machine translation system to translate texts in other languages into English: the text is translated from the original language into English, and then English-language resources such as SentiWordNet are employed [34]. Translation systems, however, have various problems, such as sparseness and noise in the data. Sometimes the translation system does not translate essential parts of a text, which can cause serious problems, possibly reducing well-formed sentences to fragments [35].

Thus, researchers look for alternative approaches. The field of multilingual sentiment analysis is now progressing very fast. In particular, multilingual lexical resources specific to sentiment analysis are being developed. For example, the NTCIR corpus of news articles in English, Chinese, and Japanese contains information on sentiment polarity and opinion holder for news related to the topics such as sport and politics [36]. However, sentiment analysis corpora and resources, even if created for multiple languages, cannot be used for other languages [37]. More research is required to improve results in the multilingual sentiment analysis discipline [38].

2.5 Current limitations

From what we have seen so far in this chapter, many issues are evident in Sentiment Analysis tasks. First of all we must consider the complexity of human language (both written or spoken). In fact, it would be too naive to oversimplify language thinking that its underlying sentiment can always be accurately examined by a machine or an algorithm. Mainly there are four main factors that currently can make sentiment analysis a complex task:

1. **Context**: a positive or negative sentiment word can have the opposite connotation depending on context.
2. **Sentiment Ambiguity**: a sentence with a positive or negative word doesn't necessarily express any sentiment although it uses the positive or negative sentiment words. Likewise, sentences without sentiment words can express sentiment too.
3. **Sarcasm**: a positive or negative sentiment word can switch sentiment if there is sarcasm in the sentence.
4. **Language**: a word can change sentiment and meaning depending on the language used. This is often seen in slang, dialects, and language variations.

A basic consideration related to the complexity of text classification can also be done considering that an automatic sentiment analysis tool's accuracy is merely the percentage of times that human judgment agrees with the tool's judgment. This degree of agreement among humans is also known as human concordance. There have been various studies in this field and they concluded that the rate of human concordance is between 70% and 79%.Taking that into consideration, we can safely say that a good accuracy for sentiment analysis tools is around 70%. The problem is: a tool that is accurate less than 70% of the time is not accurate, and a "perfect" tool that is accurate 100% of the time will draw data that we disagree with roughly 30% of the time.

Last, mainly tools for Sentiment Analysis task are basically provided in English language, but we have also discussed that some important features can be virtually lost if we do not deal with the language dependant features.

2.6 Conclusions

In this chapter we have provided a formal definition of Sentiment Analysis, a taxonomy of the the existing approaches in literature and its related works and a review of the current state of the art. Our analysis mainly focused on two distinct but fundamental aspects of the process: the features selection and the classification approaches, mainly from a machine learning perspective.

From the literature review is almost clear that the approaches towards this task, based on machine learning, are manly based on supervised techniques together with Natural Language Processing methods. There is an evident lack in experiencing totally unsupervised methods, always based on machine learning, in general Sentiment Analysis problems. In this thesis we will try to to partially fill this gap, experimenting the application of unsupervised neural networks to Sentiment Analysis, on the specific problem of cyber bulling, that will be treated as a particular case of polarity detection.

It is also clear from this chapter that there is a lack of works and resources in specific languages other than English, for Sentiment Analysis tasks, and so in the next part of this thesis, we will also propose, in the generic case of polarity detection, a model applied to the Italian language.

Chapter 3

The proposed model

"A person who never made a mistake
never tried anything new."

- Albert Einstein

Parts of this section were originally published as Di Capua M., Petrosino A. "*Semi supervised Deep Learning Approach to Deal with Data Uncertainty in Sentiment Analysis*". In Proceedings of WILF 2016, 11th International Workshop on Fuzzy Logic and Applications. LNAI 10147. Naples, Italy. 2016.

In this chapter we will introduce our proposed neural network model, providing details on its motivations and implementation. Our proposed solution will also go through the next fourth chapter with a proposed brand new model for cyber bullying automatic detection. In summary, starting within this chapter (and going through the next one), we will propose two different approaches (semi-supervised and unsupervised) to solve different Sentiment Analysis task.

3.1 Rationale

In the previous chapter we introduced recent approaches to Sentiment Analysis tasks, mainly from a machine learning perspective. We also saw that generally pattern classification tasks can be roughly divided into two groups: the so-called **supervised** methods and the **unsupervised** methods. In supervised learning, the class labels in the dataset, which is used to build the classification model, are a priori known. In contrast, unsupervised learning tasks deal with unlabeled dataset, and the classes have to be a posteriori inferred from the unstructured dataset. Typically, unsupervised learning employs a clustering technique in order to group the unlabeled samples based on certain similarity (or distance) measures. But recently another approach, that we can found in the middle, is the so called **semi-supervised** approach [39], that represents a class of supervised learning tasks and techniques that make use of unlabeled data for training, using typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data).

In this work we want to focus on both the semi-supervised approach and the unsupervised approach, due to the motivations that we don't want to put as a constraint to our solution the availability of huge pre-labelled dataset, nor we want to spend effort in labelling data, also because our field of application will be social networks, in which the amount of data make practically unfeasible this labelling activity. In addition, we also want to avoid imposing also any a priori assumption about possible classes.

In this thesis these two approaches will address two different problems in the Sentiment Analysis area. The semi supervised approach will deal with classical polarity detection in sentences. The second approach, completely unsupervised, will aim to detect automatically cyber bullying traces in social networks, a problem that we will treat as a particular Sentiment Analysis task. The details about text representation in input to the proposed neural networks and the adopted Natural Language techniques, for both of these two approaches, will be discussed later in chapter V.

We can now depict a specific experimental scheme that can be applied to classification tasks, to better decouple the various macro steps involved in the process, and to better explain our proposed solutions (see fig. 3.1).

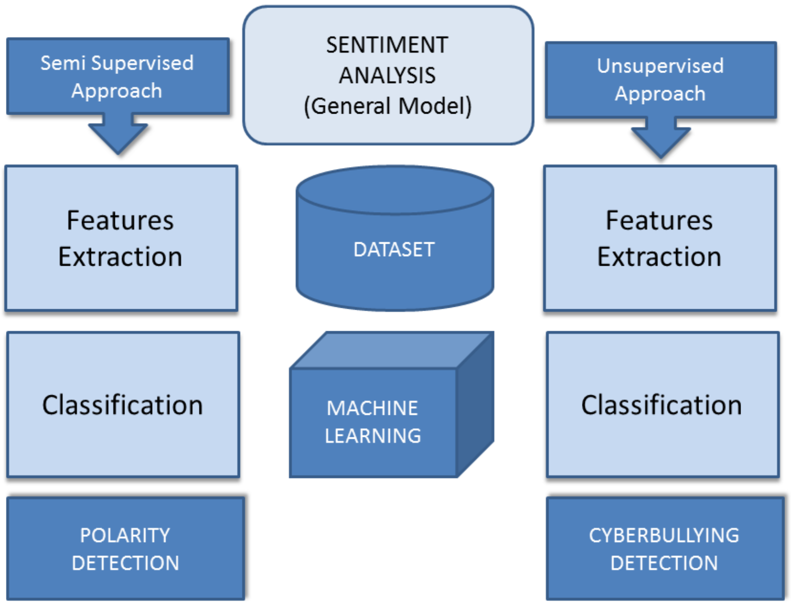


Figure 3.1: Experimental Sentiment Analysis architectural scheme for polarity and cyber bullying detection.

A fundamental first step, in a classification process, is given by the features extraction activity, in which we are interested in retaining only those features that are meaningful to our context, i.e. features that can help to build a good classifier for our problem.

At this stage of the process, it's also worth to note that we need to adopt some techniques to produce a much more clean dataset, removing some noise inside texts. Some of these activities were: removal of very frequent terms such as conjunctions and prepositions, removal of infrequent terms, removal of raw data untreatable or not significant as a URL, email addresses, etc., removal of documents (sentences, paragraphs, etc.) that appeared several times in the dataset.

For the classification stage, we just discuss how previously traditional machine learning algorithms (like SVM) have shown good performance in various NLP tasks for the past few decades, but we also find that there were some shortcomings with traditional approaches and deep learning models have the potential to overcome these limitations to a large extent. Some of the advantages of deep neural networks are:

- A strength of the deep learning models is no demand for carefully optimized hand-crafted features. Instead of the features, they take word embeddings as input which contain context information, and the intermediate layers of the neural network learn the features during the training phase itself. This means that a necessity, which is at the base of the traditional classification models, is no longer required for the functioning of deep learning models.
- Deep learning allows good representation learning. While feature representations of input text can be learned automatically from the training data for a

particular task, the representation of the words, containing context information, can be learned from raw corpus in an unsupervised manner. This disregards any need for manual construction of appropriate features or word information.

When a specific problem of artificial intelligence needs to be solved, for example tasks linked to natural language processing, often, following an intuitive approach, the problem is divided in sub-problems adopting multiple layers to represent the information [40].

Deep learning approaches aims are to learn in an automatic manner, multiple hierarchy of high level features, starting from composition of low level features. Automatic features learning on multiple abstraction layers allows the system to learn more complex (non-linear) functions mapping input and output, $f(x) = y$, starting from raw data, in this way it is possible to not depends completely on handcrafted features. This is very important for abstractions and concepts of high level for which it is not simple to specify a problem with simply binary or continue input. The ability to learn automatically which features are important, becomes fundamental when data have a very high dimensionality.

The depth of a neural network architecture refers to the number of composition level for non linear operation for the problem that it is learning. Although most of the machine learning algorithms in literature use an architecture called *shallow* (not deep), the mammal brain biological structure suggest of a *deep* architecture (layered), with an input represented with various abstraction layer, on which each single layer matches with a different area of cerebral cortex. This mechanism is particularly clear observing visual system of a primate, where there are many steps linked with visualization, which starts from detecting lines and primitive shapes and, gradually, to more complex shapes. Inspired by biological deep architecture of human brain, researcher, that worked on neural network, have attempted, for decades, to realize an efficient learning algorithm for multi layered networks, but without obtain relevant results. Generally unique positive results were relative to experiments on networks with two or three levels, but increasing network depth results tend to worsen. In 2006, G. Hinton introduces Deep Belief Networks (DBNs) that use a learning algorithm that in a greedy manner trains a layer at time. Hinton proposes to use, for each deep network layer, a specific neural network type called Restricted Boltzmann Machines (RBMs), trained using an unsupervised algorithm. Results of DBNs seems to be promising and generate a new interests towards this kind of neural networks. Deep networks are used and tested in various fields, both for classification tasks, and in fields of “regression”, “dimensionality reduction” and natural language processing and relative sentiment analysis. Although this kind of networks can be trained also with unlabeled data, they are used with success also to build deep feed-forward networks, trained also in supervised manner.

So, in our experimental scheme, for the semi-supervised approach, both the features extraction task and the classification task will be covered by a Deep Belief Network, that will be used to classify sentences in a general polarity detection problem. In the next paragraphs we will explain our DBN in details.

Instead, for the unsupervised approach, in order to detect classify cyber bullying sentences (i.e sentences with an extremely negative polarity) we propose a set of hand crafted features, that models this behaviour, and a Growing Hierachical Self Organizing Map as classifier. This approach will be covered in details in the next chapter.

In fig. 3.2 it's reported this final instantiated scheme.

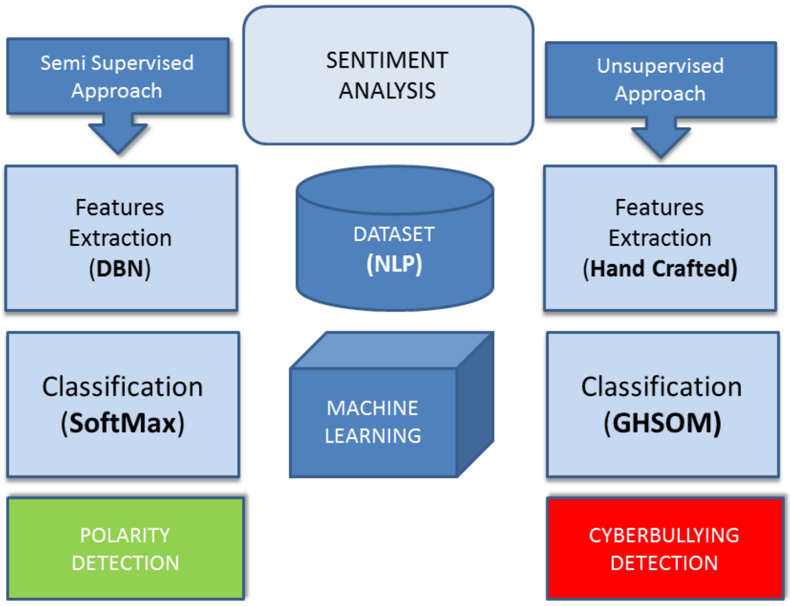


Figure 3.2: Instantiated Sentiment Analysis architectural model for our specific tasks.

3.2 A general processing architecture

The realization of the whole architectural solution is composed of several processing steps that define a processing pipeline that will be detailed here, evaluating the each individual steps from a qualitative point of view (see fig. 3.3).

In order to achieve efficient representation of the text used by a neural network, it was necessary to study and subsequently adopt an algorithm able to convert sentences in numerical values (i.e vectors in our case), that pushed semantic properties of text, necessary for the correct realization of a Sentiment Analysis solution. Our choice has been Word2Vec, the recent algorithm proposed by Mikolov et al. [4] in 2013, that will be discussed in details in chapter 5. We just recall here that our objective to deal with Italian language as the reference language, has prevented us from using exiting literature corpus and related vector representations, that are almost done in English. We then select a representative Italian corpus (Paisá from CNR) that was compatible with context, to be used with Word2Vec algorithm, in order to produce an effective vector representation of Italian terms. Considering the size of the corpus (about 250 millions of words), we obtained a vocabulary of about 250.000 Italian terms.

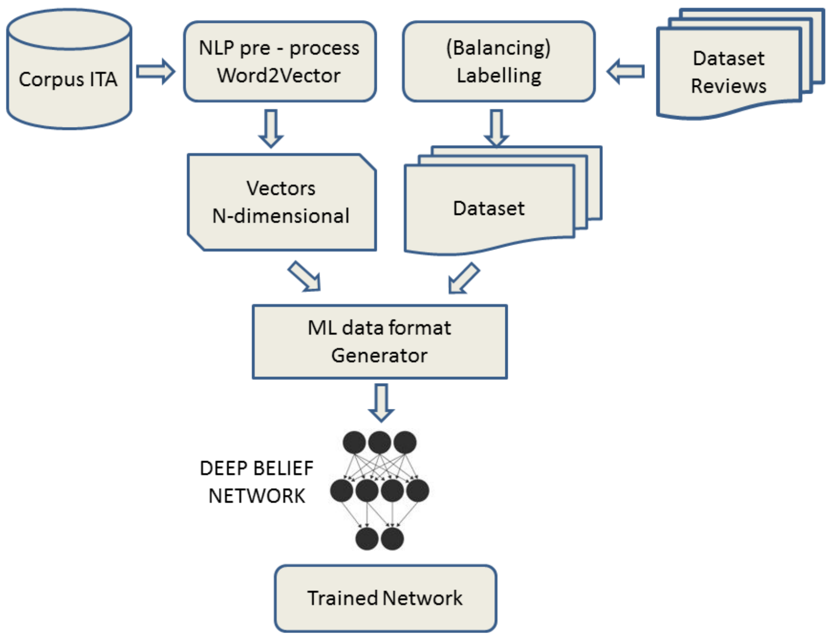


Figure 3.3: Architecture of the data pre-processing stage.

Once obtained a cleaned corpus it was possible to apply the Word2Vec algorithm to obtain a vector representations of the terms on the generated dictionary. We tested different size for the Word2Vec output vectors, also considering some hints from literature. More specifically, some representations of text inputs were produced with vectors size $K = 100, 200, 300$ and then compared in terms of efficacy. The best performance of representation of the data, verified through the semantic evaluation of clusters of similar terms, indicates that a size K = 300 (for Word2Vec output size), allows to obtain an effective representation of the terms (see chapter V for more details).

We will cover the dataset details in chapter VI, but here we can report that as sentences input to the network we take some different movie reviews taken from dedicated web portals, in Italian language. We needed also to balance these datasets, in order to achieve a better training of the network. Last, before submitting input data to the deep network, we built a simple software component to transform cleaned and balanced textual data in a vectorized representation, that can be processed by the DBN.

3.3 Intermediate representation and abstraction

Since a deep architecture can be seen as a composition of multiple computational layers, many spontaneous questions may emerge: “Which kind of data representation should be generated as output on each layer?, Which kind of interface should be used between two layers?”. Algorithms used on each layer of a deep network can be seen as focused to transform a representation of input data in a new representation of output, that will be fed into next layer and that tends to split and to bring out latent relationship underlying data. It has been observed that once obtained a good representation of data at each level, that representation can be used to initialize and, subsequently, to train next network layers, using a supervised algorithm based on the classic gradient rule. From a biological point of view, each layer of abstraction, that is in the human brain, consists of an activation phase (neural excitation), only of a small subset of a greater number of features, that typically aren't mutually exclusive[1]. Since these features are not mutually exclusive, this provide the so-called "distributed representation", and as a consequence, the information isn't localized in a single neuron, but distributed on multiple neurons. In addition to the fact that information is distributed, it seems that human brain uses also a *sparse* information representation: only a low percentage (about 1-4%) of neurons are activated at a certain time. Summarizing, fundamental requirements that a learning algorithm should respect are:

- Ability to learn complex function with an high grade of variability, that is with a number of variations larger than number of available learning example.
- Ability to learn high abstraction layer starting from a scratch input, in the way to be used to represent complex functions.
- Ability to learn from large dataset, with a scalable computational capacity (possibly in linear manner

[1]Two events A and B are mutually exclusive if it is not possible to check them at same time, that is they can't be both true. Example: "obtain 1" and "obtain 6" from a unique dice throw. In statistics this is expressed also from the fact that $P(A+B) = P(A)+P(B)$. Instead the probability that two events are not mutually exclusive is given from the sum of probability of each one less the probability that both occur, $P(AorB) = P(A) + P(B) - P(AandB))/ = P(A) + p(B) - P(A|B)$. For example the probability to have a number less or equal to three or a pair number on a dice roll is: $P(\leq 3orpair) = P(\leq) + P(pair) - P(\leq 3andpair) = \frac{3}{6} + \frac{3}{6} - \frac{1}{6} = \frac{5}{6}$.

respect to data).

- Ability to learn from data that are, mostly, unlabeled, in the way to work with semi-supervised approaches where not all training samples are provided of semantic labels.
- Capacity to capture the most part of statistical structures belong to observed samples in input (unsupervised learning).

3.4 Deep Belief Network structure

Conventional neural networks and the relative learning algorithms tend to have optimization problems based on the possibility to be trapped in local minima of objective function, resulting in a poor performance of network. Furthermore, classic neural networks doesn't take advantages from using unlabeled data, that are, often, plenty availables (such as in the case of Big Data). To relieve these problems, in the 2006 Hinton [41] introduced DBNs networks, that uses a deep architecture that is in capable to learn features representation from labeled and unlabeled data. This kind of network integrates both a unsupervised learning step, and a strategy based on supervised fine-tuning, to build a more robust and efficient model. Unsupervised step is used to learn data distribution without a priori knowledge, instead supervised step execute a local search to obtain an optimization of results.

In figure 3.4 is showed a typical architecture for a DBN, that is composed of a stack of Restricted Boltzmann Machines, and an additional layer dedicated to specific task, such as classification. Boltzmann machines are considered to be "generative" probabilistic models, that are capable to learn joint probability distribution[2] of the observed input samples, without using labels. These models are capable to use efficiently large quantity of non-supervised data.

Before of the fine-tuning phase, it is executed a pre-training process for each single layer, that is for each single RBM. Output of a RBM is used as input data for next RBM and this process is repeated until each single RBM is correctly trained. This unsupervised learning process is a crucial factor for DBN networks, that permit, practically, to avoid problems with "local optima" and relieve the "overfitting" that is easy to have when there are thousands parameters. Further more, the algorithm is very efficient in relationship to the temporal complexity; the complexity increases in linear manner respect to the number and dimension of each single RBM. Features that are represented at different layers of the architecture, correspond at different layers in terms of data structure. Note that the number of RBMS stacked to build a DBN is a predefined parameter of the architecture and the initial training step requires exclusively unlabeled data. To have a clear information of how the DBN works it is worth to describe in detail how the Boltzmann Machine works and also to describe its evolution for practical uses called Restricted Boltzmann Machine, that represents the fundamental behavior.

3.4.1 Boltzmann Machine

Boltzmann machines are energy-based neural network models, that instead to associate a probability to each configuration of system variables, associate an energy

[2]In probability, given two aleatory variables X and Y, defined on the same probability space, it is defined their **joint distribution** as the probability distribution associated to vector (X, Y)

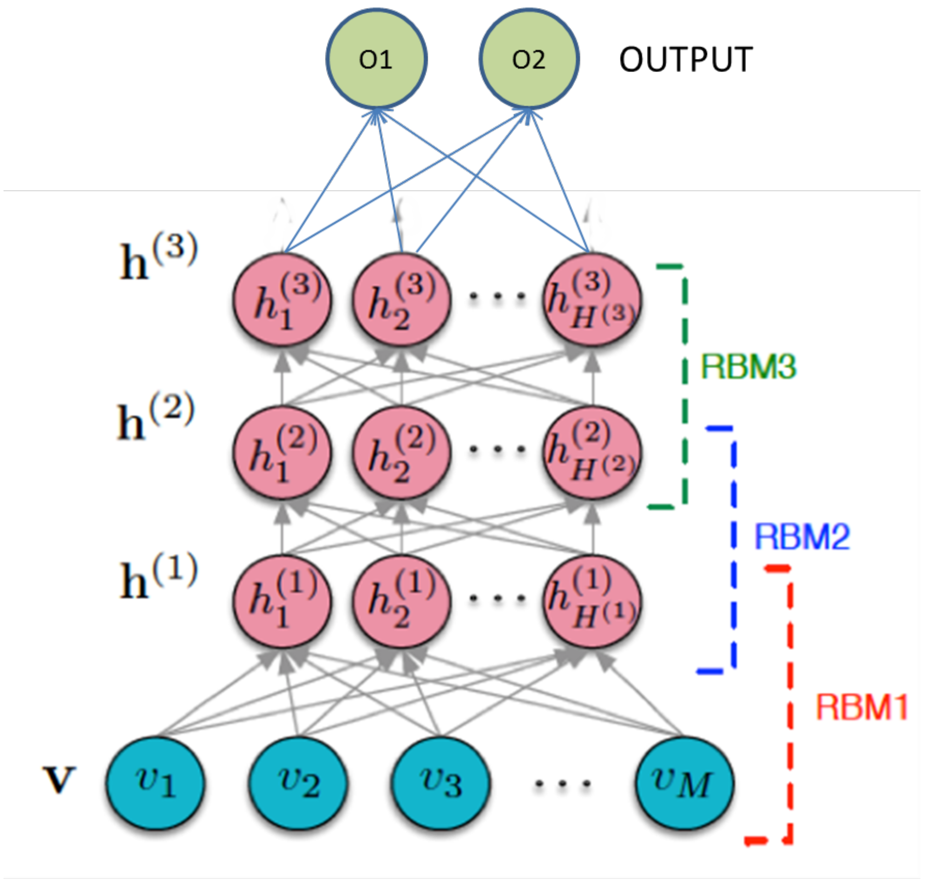


Figure 3.4: Example of a Deep Belief Network.

function removing, in this way, the necessity of a real normalization, typical of probability distribution. In this models inference problems consist on compare the energy at different configurations and selecting which assumes the desired features (minimum, maximum, etc.). Boltzmann machines are energy based models that have visible units, and hidden units, that are connected each others by weighted edges. In this type of model the main problem is to perform the training, that is try to find values for connection weights and biases of units to maximize the probability of data that we are considering. Perform this operation requires to solve an inference problem to assign values to hidden units. This problem is very complex for the high number of calculations that are necessary to repeat in order to achieve balanced values in an equilibrium state. In fact, from a structural point of view, a Boltzmann machine is a network with simple units linked together by weighted edges, which units by stochastic (random) operations are able to determine their state, which can take values between 0 and 1. Therefore, Boltzmann machines are probabilistic graphical models with the particularity to discover features and regularity of training data used as network input. Algorithms that allow learning of these features are generally very slow if the network has many units (and consequently many connections).

In a Boltzmann machine, a single unit state a_i depends from its input z_i given by the weighted summation of all units linked to them plus its bias b_i:

$$z_i = b_i + \sum_j a_j w_{ij} \tag{3.1}$$

where a_j is the state of unit j, while w_{ij} is the weight of connection between unit i and unit j. In the BM the probability that the unit i has a new active state (value 1) is:

$$P(a_i = 1) = \frac{1}{1 + e^{z_i}} = \sigma(z_i) \tag{3.2}$$

where $\sigma()$ is the logistic function, a common sigmoid function.

To describe the learning process of a BM it is common to use a “thermodynamic” metaphor: the goal is to reach a new thermal equilibrium in the network, which corresponds to a distribution called Boltzmann[3], where the probability that the vector state of visible units v is determined by energy function. The energy function is defined as:

$$E(v) = -\sum_i a_i^v b_i - \sum_{i<j} a_i^v w_{ij} \tag{3.3}$$

where a_i^v is the state of unit i assigned to vector v. Then it is possible to define $P(v)$:

$$P(v) = \frac{e^{-E(v)}}{\sum_u e^{-E(u)}} \tag{3.4}$$

The denominator indicates the sum of all units in the network.

To better explain how the learning algorithm works in an BM let's consider all variables to be visible. To perform the network training, given the training set, it is necessary to search weights and biases to maximize the probability that each vector v is contained in the dataset. More accurately we search weights and biases that allow to reach a Boltzmann distribution, where training set vectors have a high probability. To search that values (weights and biases) it is possible to use an approach based on descendant gradient[4]. Using a gradient is due to not regular trends of the function to maximize, which can have a lot of local maxima, making impossible to use

[3]In thermodynamic the average kinetic speed of a system (for example particles) is proportional to the temperature of the system itself.

[4]Descendant gradient is a local optimization technique. Given a mathematical function with multiple variables, gradient descend allows to find a local minima of function. It consists in evaluating, from an arbitrary starting point in the multidimensional space (first point), the function itself and its gradient. It following a descent direction, indicates the direction where function tends to minimum. After another point is chosen (second point) in the direction indicated from gradient. If the function, in the second point, has a value less than the value in the first point, descent can continue, following gradient at second point, which could be very different from the previous one.

an exact search method. Starting from the definition of P(v) and with knowledge that it is possible to write:

$$\frac{\partial E(v)}{\partial w_{ij}} = -a_i^v a_j^i \tag{3.5}$$

we can also write:

$$\left\langle \frac{\partial \log P(v)}{\partial w_{ij}} \right\rangle data = \langle a_i a_j \rangle \, data - \langle a_i a_j \rangle \, model \tag{3.6}$$

where the operator $\langle \cdot \rangle \, data$ indicates expected values given using sampling of BM, after the equilibrium state is reached. Therefore to apply the gradient we should generate examples from BM, after the equilibrium has been reached, obtaining the values of right side of equation, and going to modify weights w_{ij} based on the following difference:

$$\Delta w_{ij} = \eta(\langle a_i a_j \rangle \, data - \langle a_i a_j \rangle \, model) \tag{3.7}$$

where η is the learning rate. If we now add the distinction between hidden and visible variables, learning rules don't change, but now to compute $\langle a_i a_j \rangle \, data$ we have fixed visible variables, and to obtain the hidden variables it is necessary to continually recompute them (using visible variables constraints) until equilibrium is reached. This continuous calculus of hidden variable, respect to all the other variables connected to this variable is called Gibbs sampling. This operation requires a high number of calculus to reach the equilibrium, that makes this kind of networks less useful for real scenarios. A variant of BM that solves this computational complexity is called Restricted Boltzmann Machine.

3.4.2 Restricted Boltzmann Machine

A solution to this problem comes from Restricted Boltzmann Machines with a structural modification of the network. It is forced the presence of two distinct set of neurons to form a bipartite graph, that doesn't allow connections within the same set. The two set of variables are those made of visible variables and those made of hidden variables. In this way the definition of the conditional probability of one and the other set of variables results to become factored, and it is possible to define an algorithm to approximate in an efficient manner the calculus of weights and biases of the network. This algorithm is called **contrastive divergence**, and it will be described later.

Restricted Boltzmann Machines are effectively used for dimensionality reduction task, classification, regression, learning and topic modelling. These models are fundamental to build particular deep models as the Deep Belief Network, obtained stacking many RBMs. An RBM is a two layer network: the first layer is called “visible” or “input” layer, while the second layer is called “hidden”.

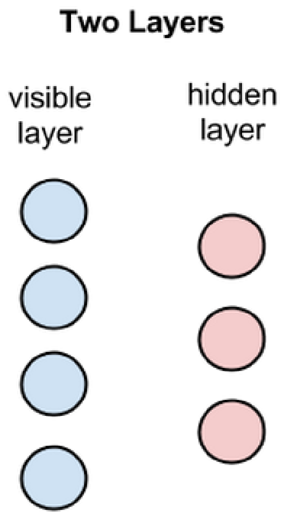


Figure 3.5: Visible and hidden layers of a RBM.

In figure 3.5 each "circle" represents a neuron of the network called node. These units are the elements where the computation is performed. Neurons are connected between them only if belonging to different layers, and therefore connections between neurons in same layer are not allowed. This lack of layer interconnections is the fundamental element that defines Boltzmann machine as restricted. Specifically a RBM sends input data to all nodes of its hidden layer, and, for this reason, it can be defined as a bipartite symmetric graph. Symmetry and bi-partition allow each node to be completely connected with the other layer.

Each input node computes input data following stochastic processes[5] to check if these data should be transmitted to next layer. For each data a specific set of features is taken into account, for example in an image is used the intensity of the pixel, or in case of text the frequency of a

[5]A stochastic process is an ordered set of random variables, indexed by a parameter t, usually defined as time.

word in a document. These numeric values, when transmitted to the hidden layer, are multiplied for a weight value and then summed to a second value defined as the bias, which help to regularize values for next operations. Result is sent to a function called “activation function” that produces the output value learned from the previous data.

$$activationf((weight * data) + bias) = output \quad (3.8)$$

Figure 3.6 shows how a single node works for each input data. Instead figure 3.7 explains how a single input is processed from all nodes of the visible layer and then sent to hidden layer. In these examples data are composed of four features, and each of them is computed by a single node, multiplied for the specific weight and afterwards sent to next layer which finally produces the output.

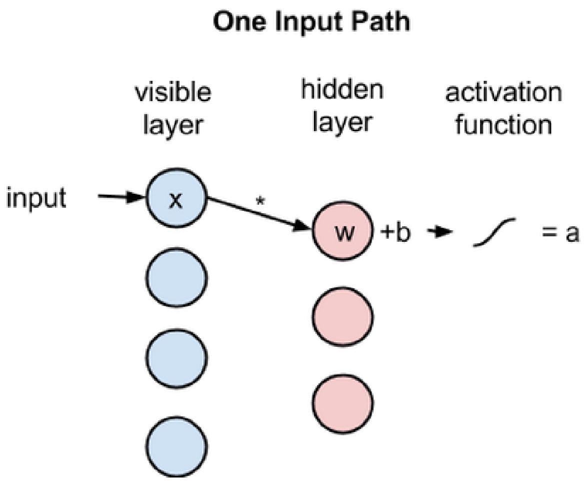


Figure 3.6: Single process flow for each node.

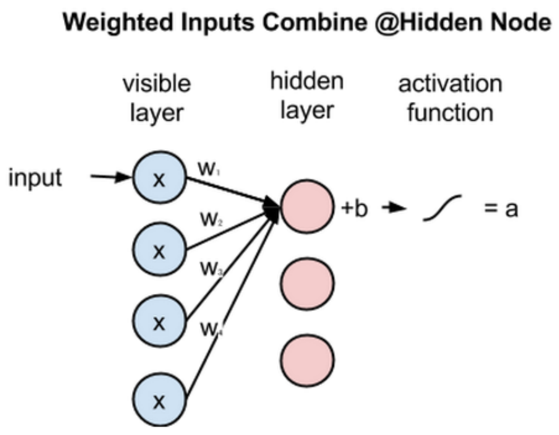


Figure 3.7: Feeding of a single hidden node from the visible layer.

It is also possible to build easily a "deeper" network than a single RBM, simply stacking more RBMs and using the output of a network as the input of the next network. This approach can be extended using many layers, generating a network called "deep", and adding as a top layer a classification one, as showed in figure 3.8

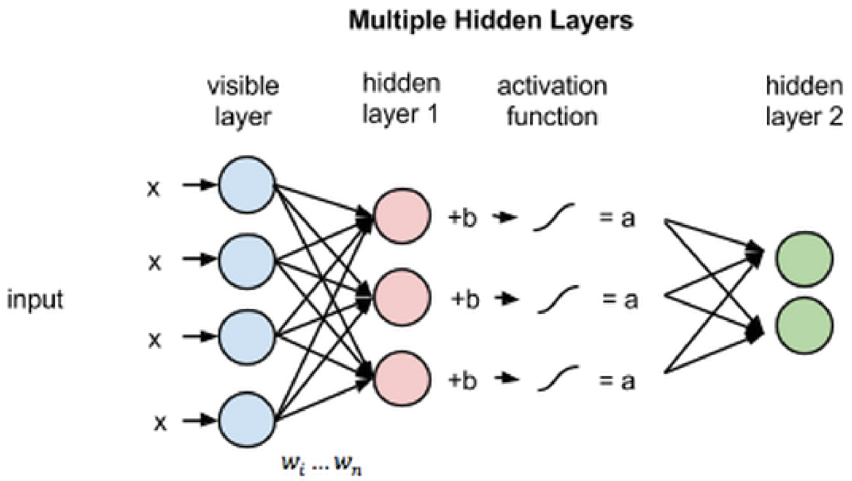


Figure 3.8: Deep network with stacked RBMs.

3.4.3 Reconstruction phase

Thanks to RBMs it is possible to reconstruct data starting from data itself, with an approach defined as unsupervised, without using a priori knowledge of the final result, but allowing network to learn it. Coming back to the easier problem of dealing with a network made of a single layer (input / hidden) it is possible to perform many forward steps and backward (backpropagation) steps, allowing the output to be recomputed from previous layers, in order to minimize the learning error.

This step is called reconstruction. When output is sent back to the input layer from the hidden layer it is applied again a multiplication for weights connected between two levels and then summed, exactly like in the first step, except for the activation function that is not used.

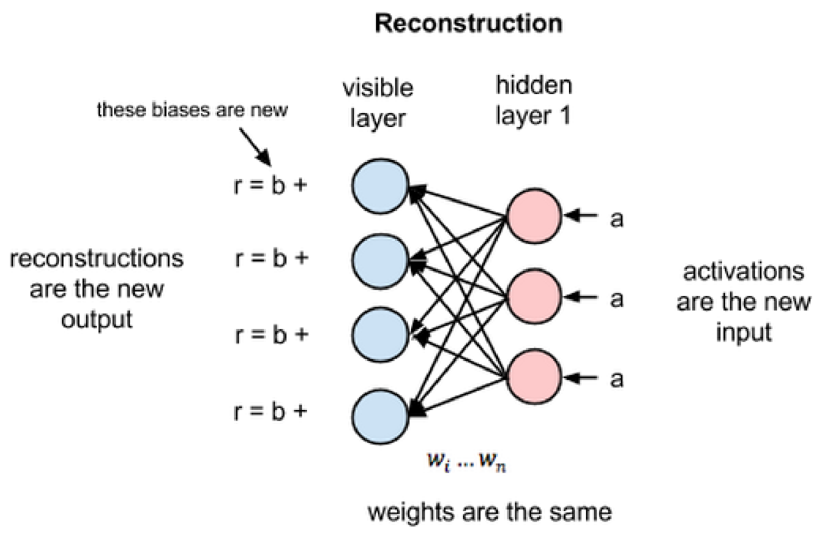


Figure 3.9: Reconstruction step.

Usually the initial difference between the two steps of the algorithm is large, because data are initialized in a random manner. This step (forward, backward) is performed many times to minimize the difference among weights.

3.4.4 Probability distribution

As it is possible to observe from the “forward” training step, the RBM uses inputs to compute a probability estimation of output given a weight x, that can be seen as the conditional probability of result a given the weighted input x:

$$p(a|x; w) \tag{3.9}$$

Vice versa in the backpropagation step the RBM tries to estimate the probability of input x given the result a:

$$p(x|a; w) \tag{3.10}$$

These two probabilities mixed together give the joint probability of the input x and the output a:

$$p(x, a) \tag{3.11}$$

In this way we don’t try to find a class to assign to a single input, but we try to extract the input probability distribution. This learning process is called *generative learning*, which is different from the *discriminative learning* where pure classifiers try to separate data in distinct set.

Once defined the learning process as the determination of two probability distribution, one for input and one for output, what allows us to maximize the joint probability is to search the distance between the two distribution and minimize it. This operation can be computed using *the Kullback Leibler Divergence* (KL-Divergence), which measures the not overlayed area (divergence area) between the two distributions seen as two different curves. The RBM tends to reduce this distance at each step. In figure 3.10 on left we can observe the probability distribution of input $p(x)$ and of the reconstruction $q(x)$ at a generic step of the RBM, on right side instead it is possible to observe the divergence area obtained with the KL-Divergence.

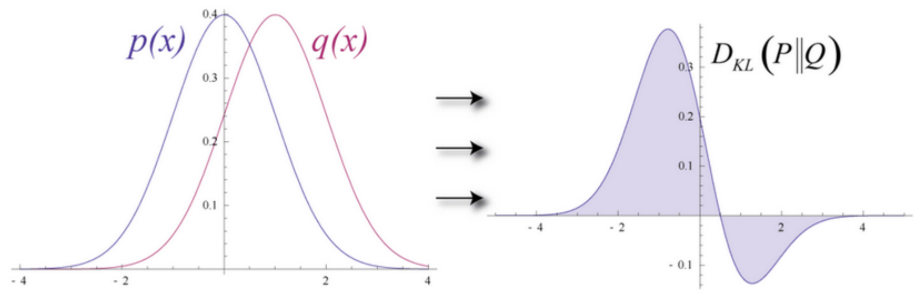


Figure 3.10: Divergence area of KL-Divergence.

Running continuously, the RBM weights are adjusted at each step reducing from time to time the total error up to approximate as much as possible the input data, as shown in figure 3.11.

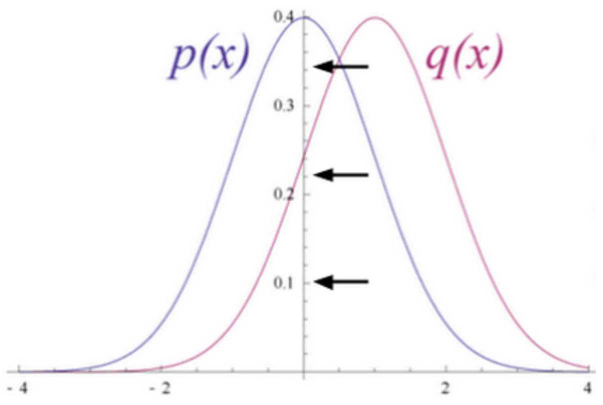


Figure 3.11: Approximation of reconstruction $q(x)$ of the input $p(x)$ at each run of the RBM.

3.4.5 Contrastive divergence

We have seen how the fundamental problem to train a "generative" network, like an RBM, is to find the probability of a function $f(z)$ respect to a probability distribution $p(z)$. Obtaining a set of samples $z_l, l = 1...L$, which are independent of each other, from $p(z)$, the probability can be seen as the approximation $p_{approx}(z)$. To obtain these samples practically it is possible to bring a Markov chain to convergence.

Referring to the KL-divergence described previously and considering P_0 referring to input data distribution, while P_∞ referring to the probability distribution of the output (equilibrium state), the problem can be reworded as:

$$KL(P_0||P_\infty) = \sum P_0 \log(P_0) - \sum P_0 \log(P_\infty) \quad (3.12)$$

In this formula, while it is simple to evaluate the term $\sum P_0 \log(P_0)$, which represents the training of data points, performing a sampling of $\sum P_0 \log(P_\infty)$ requires instead a Gibbs[6] chain potentially of infinite length.

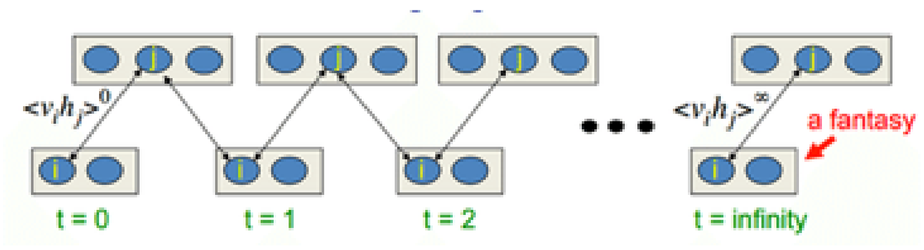


Figure 3.12: Reconstruction (potentially infinite) for a RBM, using Gibbs sampling.

Due to the fact that this operation can be very expensive, it is possible to use a fewer number of steps of the Markov chain than those required to obtain an approximation of the probability distribution $p_{approx}(z)$. This result can be obtained using a technique called *Constrastive Divergence* (CD) proposed by Hinton and composed by the following steps:

1. The Markov chain is initialized by training data (which have a distribution expected to be close to p).
2. Reaching convergence is not required, but k sampling steps are performed, using Gibbs sampling.

Hinton proposes to minimize the difference between $KL(P_0||P_\infty)$ and $KL(P_1||P_\infty)$, where P_1 is the distribution obtained after only one sampling step (CD-1). For this reason computing the difference between the two divergence doesn't require to evaluate $\sum P_0 \log(P_\infty)$, but only the term $\sum P_0 \log(P_1)$, using only one, but sufficient, Gibbs step. The motivation of the effectiveness of this approach, is given by the fact that the distribution

[6]In statistics, a Gibbs sampling is an algorithm based on Monte Carlo Markov Chain used to obtain a sequence of random samples from a multivariate probability distribution (that is the joint probability distribution of two or more random variables), when direct sampling is hard to perform. This sequence can be used to approximate the joint distribution (for example to generate an histogram of distribution); to approximate the marginal distribution of one of the variables, or of various variables subsets (for example, unknown parameters or latent variables); or to compute an integral (such as the expected value of one of the variables).

P_1 is much "closer" to the distribution P_∞ with respect to the distribution P_0, the difference of the two divergences is always more than zero. Therefore minimizing this quantity on more step has the same effect as to minimize the original divergence between P_0 and P_∞.

Generally we can summarize that CD is a method for non-directional graphical models training (a class of probabilistic models used in machine learning). Based on the approximation of the gradient of the log-likelihood (basic criterion which the most of probabilistic learning algorithms try to optimize) using a short Markov Chain (a method to sample from probabilistic models) starting from the last observation.

3.5 Multiple layers definition

After the structure of an input data has been learned, it is in the hidden layer abstraction state. At this point it is possible to pass the output data to a second hidden layer, using the first layer as a visible layer. In this way a second set of output data is produced, generating a second set of features in a hierarchical fashion, which gives the network the opportunity to learn a more complex and abstract data representation. Inside each new layer the learning process is repeated until the error is minimized and then the output is transferred to a new layer, until the last layer is reached. This approach permits to avoid data modification in order to adapt them to the network, and also to avoid providing a pre-classification step of data.

3.6 Hyper-parameters tuning of the model

Learning algorithms for artificial neural networks and in particular for Deep Learning involve setup of peculiar variables called ***hyper-parameters***. We have seen how the previous learning techniques are based on unsupervised models [41] that can be applied more than once to perform training steps to different layers of the deep structure. In particular deep learning algorithms, such as those based on Boltzmann machines, often provide a further step of supervised refinement. This refinement, also better known as *gradient descendant*, includes the algorithm of backpropagation, like in the “feedforward” supervised neural networks or in “recurrent” neural networks. In this paragraph will be discussed some best practices followed in this study, useful to train a deep network based on Restricted Boltzmann Machines.

The notion of reuse, which helps to explain distributed representation of data, is one of the great advantages behind Deep Learning. For example, the complex theory of circuits ([42] [43]) (which includes neural networks as a special case), has anticipated much recent researches on deep learning. The depth of a circuit is the length of the most long path from one input node of a circuit to an output node. Formally, it is possible to change the depth of a circuit changing what each node can process, but only of by a constant factor [40]. Typical elaborations allowed in each node include: weighted sum, product, artificial neuron model (as monotone non-linearity above affine transformations), kernel processing, or logical circuits. Theoretical results ([42], [43], [44] [45]) identify clearly a family of functions where a *deep* representation can be exponentially more efficient than one that is not deep enough. If the same set of functions can be represented by an architectural family, learning theory suggests that this set can be learned with a few neurons, improving computational and statistical efficiency.

Another important motivation for features learning and for deep learning is that it can be carried out with unlabeled data, as long as the random variables (not observed) are relevant to the initial data distribution. This is true under the *manifold hypothesis*, which states that natural classes and other high level concepts, in which humans are interested, can be associate with region of low dimensionality in input space (manifold space) near which distribution focuses, and different manifold classes are well separated from low density region. This means that a little semantic change around a particular example can be captured changing only few numbers in a high level representation of the space. As a consequence features learning and deep learning are strictly related with principles of unsupervised learning, and can be applied in a semi-supervised environment (where only few samples are labeled), as much as in *transfer learning* or in multi-class environments.

One of the most common used approaches to train a deep neural network is based on the concept of *greedy layer-wise pre-training* [44]. The idea, introduced by Hinton in [41], is to train a single layer of the deep architecture at time, using an unsupervised approach. Each layer takes as input the model learned in the previous layer and learn a new model from it. This model can be used as input to predict variable of interest, in order, for example, to classify objects. After the unsupervised pre-training step, it is possible to perform a supervised refinement of the whole system, for example, optimizing not only the classifier, but also underlying layers. Combining the two methodology, usually, gives a better generalization of the only supervised model using a random weight initialization. Pre-training unsupervised algorithms proposed in 2006 were based on Restricted Boltzmann Machines (RBM) [41], autoencoders [46] and a sparsification of autoencoders similar to sparse coding [47].

A learning algorithm can be seen as a function that takes as input training data and produces as output a new function (called predictor) or a model (a set of functions). However, in practice, many algorithms use hyper-parameters. We define an hyper-parameter for a learning algorithm A as a variable to set before use A on data. Practically it is an external control knob. It can be discrete (as in the model selection) or continuous (as in the selection of the learning rate). In many algorithms, such as in deep models, there is an high number of hyper-parameters (that can be also greater than 10), and it is necessary to setup them in the right way, to obtain good performances. Choosing the value of a set of hyper-parameters is equivalent to select a learning model. Therefore the question is: given a set of algorithms, how to choose the most appropriate configuration of hyper-parameters?

We will focus on which algorithms are relevant to neural networks and deep learning. Many learning algorithms can be seen like a combination of two elements:

1. A learning criterion and a model (a family of functions, a parametrization);
2. A procedure for criterion optimization;

We report below a definition of the most important hyper-parameters, associated to learning algorithms:

- **Initial learning rate (ϵ_0).** Often this is the most important hyper-parameter and it is fundamental that we try to set it to the best possible value. Typical values for a neural network with a standardized input (i.e. in the range [0,1]) are less than 1 and greater than 10^{-6}. A standard value of 0.01 typically works fine for multilayer neural networks, but it is important to not take these values as an absolute truth, but only as an initial reference because everything depends on the model parametrization. If, for example, there is time to optimize only an hyper-parameter and we are using the stochastic gradient descend, then can be useful to set and fix this parameter.
- **Mini-batch size (B).** Typically this value is chosen between 1 and few hundreds, for example $B = 32$ is generally a good setting. With values greater than 10 advantages of matrix-matrix products are appreciable, with respect to matrix-vector product. Impact of B is mostly computational, big values of B allow fast computations (with appropriate implementations), but it is needed to repeat the operation many times on samples before reaching a good error minimization, because there are less updates for each iteration. Usually this parameter impacts the learning speed and not on performance of test step, than it can be optimized separately from other hyper-parameters. Comparing learning trend (curves that represent training/validation error on learning time trend) this parameter can be chosen after the settings of other hyper-parameters (excluding the learning rate). B and ϵ_0 can slightly interact with other hyper-parameters, therefore they could be optimized

again at the end of the process. After B is found, it is possible to refine again the other parameters to further improve performances.

- **Number of training steps (iterations)** T (They are measured as a mini-batch update). This hyper-parameter is particular because it can be optimized without restrictions using the *early stopping* principle: monitoring learning error (that can be estimated from a validation set) as the value of the learning progress, it is possible to decide how long to train the network for each hyper-parameter setting. The *early stopping* technique is a not expensive way to avoid overfitting. In this way even if other parameters tend to generate overfitting, early stopping reduces considerably this problem. This means, however, that it could also hide overfitting caused from other hyper-parameter settings, making vain analysis that can be done to identify the effect of each hyper-parameter. It is, therefore, a good choice to disable it during the optimization step and activate it after good refinement for all the hyper-parameters of model is achieved.
- **Number of hidden units** η_h**.** Typically you are free to choose the size of each layer of the network in order to control its capacity. Because there are adopted methodologies like early stopping and other regularization techniques, it is very important to chose a η_h big enough. A very big value, far from the optimal one, typically doesn't influence generalization ability of a network, but it requires proportionally many more computations ($O(\eta_h^2)$ if all layers are used at same time on a completely connected architecture). As it happens for many other hyper-

parameters it is possible to choose a different value of η_h for each hidden layer of a deep architecture. In an important comparative study [40] is shown that using the same size for all layers works generally better, or in the same way, as setting a decreasing size (like in pyramidal style) or an increasing size (like in inverted pyramidal style), but also in this case the setup is certainly dependent from used data. On many published works it is instead found that a first hidden layer size greater then the input data size, works better of a reduced size layer. Empirical observations show that an optimal η_h must be much bigger in cases a pre-training unsupervised approach is used to generate data for a supervised network. A plausible explanation is that many hidden layers, after pre-training steps, bring with them information not relevant for the supervised task of interest. But to be sure that relevant information for a task can be caught, it is mandatory to use hidden layers with size greater than that required.

The neural network model designed for our Sentiment Analysis activities is based on Deep Belief Networks, obtained by stacking some Restricted Boltzmann Machines. Different structural models have been tested, varying the shape of the network, or by varying the size of the input level, the number of hidden levels as well as the number of neurons of each level. In the figure 3.13 below we present a general outline of the proposed model of the Deep Belief Network.

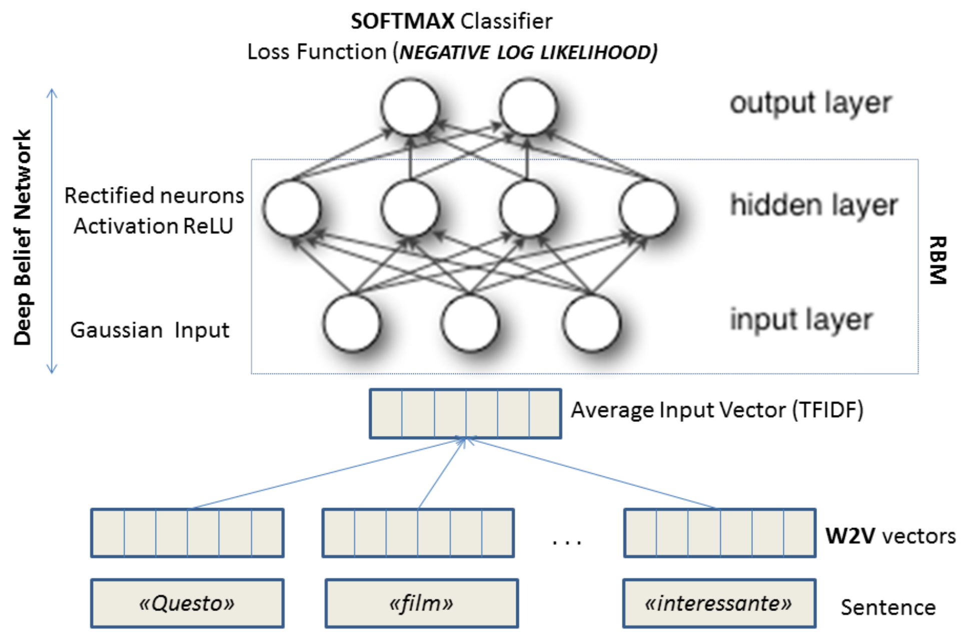


Figure 3.13: Architecture of the proposed DBN-W2V.

Some choices of our model are represented by classifier type **SoftMax**, with error function based on Negative Log Likelihood loss function. SoftMax is a function used as the output layer of a neural network that classifies input. It converts vectors into class probabilities. SoftMax normalizes the vector of scores by first exponentiating and then dividing by a constant.

At the visible unit layer, i.e. the layer of nodes where input goes in, we adopted **Gaussian neurons**, which have an activation function based on the Gaussian function.

Our activation function of the hidden layer nodes choice is the **Rectified Linear Unit** (ReLU) [48]. The ReLU function has become very popular in the last few years. It computes the function $f(x) = max(0, x)$. In other words, the activation is simply thresholded at zero (see image above on the left). There are several pros and cons to using the ReLUs [7]:

- (+) ReLU greatly accelerate the convergence of stochastic gradient descent compared to the sigmoid/tanh functions. It is argued that this is due to its linear, non-saturating form.
- (+) Compared to tanh/sigmoid neurons that involve expensive operations (exponentials, etc.), the ReLU can be implemented by simply thresholding a matrix of activations at zero.
- (-) Unfortunately, ReLU units can be fragile during training. For example, a large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any datapoint again. If this happens, then the gradient flowing through the unit will forever be zero from that point on. That is, the ReLU units can irreversibly die during training since they can get knocked off the data manifold. For example, you may find that as much as 40% of your network can be "dead" (i.e. neurons that never activate across the entire training dataset) if the learning rate is set too high. With a proper setting of the learning rate this is less frequently an issue.

From our tests we effectively saw that this function tends to create more robust activations and improves the evaluation F1 score, as will be explained in chapter VI.

The initialization of the input level weights were made through the **Xavier** algorithm, proposed by Xavier Glorot and Yoshua Bengio in [49]. In short, this kind of

[7]Andrey Karpathy's blog - http://cs231n.github.io/neural-networks-1/

initialization helps signals reach deep levels of the network. If the weights in a network start with too small values, then the signal shrinks as it passes through each layer until it is too tiny to be useful. On the contrary, if the weights in a network start too large, then the signal grows as it passes through each layer until it is too massive to be useful.

On this proposed model we tested different layers configuration and general parameters tuning of the model. More in details, we performed different experiments in order to observe how *accuracy* can change with respect to different hyper-parameters changes. Tests were also conducted on two different datasets, in order to also test the effectiveness of the model generalization with respect to different styles of comments. In the table 3.1 we reported various tests obtained varying parameters such as learning rate, the network layers structure, the number of iterations, epochs, etc. In the first column of the are reported the different DBN configurations, obtained by varying both the number of levels of the network and the number of hidden neurons of the RBM. The tests in the table 3.1 refer to the dataset A. (See chapter VI for more details on these data sets).

Table 3.1: Accuracy results respect to different network configurations hyper-parameters, tested on dataset A, with input size $k = 100$

Network topology	**Batches**	**Epochs**	**Iterations**	**Learning rate**	**Accuracy**
100-200-2	100	5	10	0.5	**0,9714**
100-200-2	400	1	50	0.5	0,8679
100-200-2	500	1	50	0.5	0,8057
100-200-2	500	2	100	0.5	0,88
100-200-2	800	7	40	0.5	0,825
100-200-2	1000	2	100	0.5	0,7914
100-200-2	1000	3	100	0.5	0,7929
100-200-2	1000	1	200	0.5	0,7829
100-200-2	1000	20	50	0.5	0,8243
100-200-2	1000	7	40	0.5	0,7971
100-200-2	1000	20	50	0.2	0,7886
100-200-2	1000	20	100	0.5	0,8129
100-200-2	2000	30	20	0.2	0,7507
100-200-2	2000	30	20	0.5	0,7629
100-200-2	2000	20	100	0.5	0,7507
100-200-2	3000	2	100	0.5	0,7057
-	-	-	-	-	-
100-300-2	100	2	10	0.2	**0,9**
100-300-2	1000	1	200	0.5	0,8229
100-300-2	1000	20	30	0.2	0,8171
100-300-2	2000	1	200	0.5	0,7643
100-300-2	2000	10	20	0.25	0,7536
100-300-2	2000	15	20	0.25	0,7629
100-300-2	2000	30	20	0.2	0,7579
-	-	-	-	-	-
100-400-2	1000	5	25	0.5	0,82
-	-	-	-	-	-
100-300-300-2	2000	15	20	0.25	0,7186

It can be observed from these results that the indications derived from the literature, on the "best practices" of the configuration can be actually confirmed by the trend of some of results shown in the above tables. For example, it is evident that, compared to the structure of the network, a number of growing neurons from input to output, to the hidden internal levels, actually appears to provide better performances.

It's worth to note that for a very small set of data

Table 3.2: Accuracy results respect to different network configurations hyper-parameters, tested on dataset A, with input size $k > 100$

Network topology	**Batches**	**Epochs**	**Iterations**	**Learning rate**	**Accuracy**
200-400-2	1000	30	10	0.5	**0,8757**
200-400-2	1000	2	10	0.5	0,8214
200-400-2	2000	5	10	0.5	0,7729
200-400-2	3000	5	10	0.4	0,741
200-400-2	3000	10	20	0.6	0,7748
200-400-2	3000	30	10	0.5	0,7643
200-400-2	4000	100	10	0.5	0,7371
200-400-2	4000	50	5	0.5	0,7404
200-400-2	4000	100	20	0.5	0,7411
-	-	-	-	-	-
300-600-2	1000	5	10	0.5	**0,8714**
300-600-2	2000	5	10	0.5	0,8357
300-600-2	2000	10	20	0.5	0,8579
300-600-2	4000	10	20	0.5	0,78
-	-	-	-	-	-
200-400-200-2	3000	5	10	0.5	0,67
200-400-400-2	3000	5	50	0.5	0,73
200-400-800-2	3000	5	10	0.7	**0,7448**
200-400-600-2	3000	2	10	0.7	0,7119
200-300-400-2	3000	5	20	0.5	0,7038
200-300-400-2	3000	10	20	0.5	0,7186
200-200-200-2	3000	10	20	0.7	0,68

(e.g. 100 documents) in the case of the data set A in table 3.1, performances that appear excellent (0,9714), are actually related to the **overfitting**[8] problem, because of the obvious disproportion between the dataset and the number of features evaluated. For this reason, in the model we also adopted some techniques that try to mitigate the overfitting, such as the so-called **regularization** (L1 and L2), a technique that adds a penalty term to the calculation of the weights, so as to avoid that the model became perfectly suited to the data. The difference between the L1 and L2 techniques consists in the fact that

[8]In statistics and computer science, the overfitting (excessive adaptation) can be observed when statistical model fits the observed data (the sample) using an excessive number of parameters. An absurd and wrong model can fit perfectly if it is fairly complex compared to the amount of available data.

Table 3.3: Accuracy results respect to different network configurations hyper-parameters, tested on dataset B, with input size $k = 300$.

Network topology	**Batches**	**Epochs**	**Iterations**	**Learning rate**	**Accuracy**
300-600-2	1000	3	10	0.5	0,9171
300-600-2	1000	3	10	0.1	0,7371
300-600-2	5000	30	5	0.4	0,7849
300-600-2	10000	5	5	0.5	0,7463
300-600-2	10000	5	10	0.5	0,758
300-600-2	10000	20	10	0.5	0,7616
300-600-2	10000	10	25	0.5	0,7599
300-600-2	10000	2	3	0.4	0,6781
300-600-2	10000	5	10	0.4	0,744
300-600-2	10000	5	10	0.2	0,737
300-150-2	12000	5	10	0.5	0,6992
300-150-2	12000	3	10	0.5	0,6971
-	-	-	-	-	-
300-300-600-2	10000	10	5	0.5	0,699
300-600-900-2	1000	10	5	0.4	0,7829
300-600-900-2	10000	10	10	0.2	0,7091
300-600-900-2	5000	3	5	0.6	0,7106

the L2 method is based on the sum of the squares of the weights, while the L1 is based only on the sum of the weights.

3.7 Conclusions

In this chapter we have introduced our proposed model based on Deep Belief Network, to accomplish a general Sentiment Analysis task, from a semi-supervised perspective, based on polarity classification of sentences. We explained the rationale of our choice, and provided our proposal and motivations for the configuration settings (hyper-parameters) of this network. We have also introduced, as input data to the network, a vectorized representation of textual sentences, but differently from classical literature on this argument, we addressed the Italian language. We reported in this chapter the general results obtained, while a detailed discussion of our experiments will be covered in chapter VI.

Chapter 4

An Unsupervised Cyber bullying Detection Model in Social Networks

"We cannot solve our problems with the same thinking we used when we created them."

- Albert Einstein

Parts of this section were originally published as Di Capua M., Di Nardo E, Petrosino A. "*Unsupervised Cyber Bullying Detection in Social Networks*". Proceedings of ICPR, 23rd International Conference on Pattern Recognition. Cancun, Mexico. 2016.

4.1 Introduction

Due to the recent growing of the cyber bullying phenomenon, there are no formal or shared definitions of what cyber bullying exactly is. Among the different definitions, one commonly used in research papers, within this area, is the one provided by Patchin and Hinduja

[50], that consider cyber bullying as the activity of ”harming or harassing someone via internet or social networks in a repeated and deliberate manner”. In social networks these activities are carried out sending messages containing harmful sentences, offending other people in front of the rest of on-line communities. With the spread of mobile technologies, cyber bullying has become an increasing problem, especially among teenagers. Awareness has also increased, due to some episodes of suicide. According to recent studies [51] almost 43% of teenagers in the U.S. revealed to be victims of cyber bullying. It is, therefore, evident that the availability of tools that can automatically identify possible behaviors classified as cyber bullying, can be really useful to prevent situations of “risk” to the victim. Even if the problem is now heavily considered from a social point of view, computational studies in this field are largely yet unexplored and only few researches on cyber bullying are available. We propose a possible solution for automatic detection of the bully traces, i.e. social media posts containing harmful text or sentence that could possibly lead to a cyber bullying episode. We shall show that using both techniques derived from NLP, the pre-processing data stage, and the subsequent adoption of machine learning algorithms, for the detection phase, can lead to reliable results. We propose here a brand new model of a cyber bullying detection on the basis of Sentiment Analysis approach, considering, as an assumption, that a cyber bullying post is an extremely negative message. The unsupervised approach we pursue is aimed to avoid manual labelling of the datasets that are huge and imposing any a priori assumption about possible classes. In the following sections, after a background introduction and a view

on the state of the art in this area, we present our cyber bullying detection model and we present some results obtained from applying our model to some dataset.

4.2 Background

Research in sociology and in psychiatry can provide important algorithmic insights to define models in order to detect bully traces, i.e. candidate instances of cyber bullying. A cyber bully status has been shown to be associated with several problems as hyperactivity, bad conduct, low pro-social behavior, but also with smoking and drunkenness [52]. From a psychological point of view there are some special features that differ cyber bullying from the traditional bully behavior. First, the absence of relationship between victim and bully. In fact, for those who suffer harassment is even more difficult to defend themselves against this phenomenon because very often the victims cannot even identify who the bully is. Often the bully hides himself behind false names and not having direct contact with the victim lowers his inhibition. The aggressor does not always receive communications by the victim that may mitigate or modify his aggressive behavior, and also many victims feel helpless because they cannot identify and then respond appropriately to their aggressor [53]. Last, the lack of space and time limits represent another fundamental aspect: cyber bullying can damage the privacy of the victim, at any time of day or night. Given the characteristics of virtual communication it is also necessary to reconsider the criterion of repetition inside cyber bullying phenomenon; in fact, only one simple information (message, video, or a photo) disclosed to many people through the Internet or smart phones, can cause damage to the victim regardless of its repetition, being able to be viewed and re-transmitted by many people at different times. The temporal dimension of the offense hypothetically can expand indefinitely, because over the web the information remains available to the community for a long time regardless of the future actions that actively the cyber bully can do. Strictly speaking, in cyber bullying is not necessary that the offensive act is repeated over time by the same person to cause a personal damage. We can conclude the background examination introducing a useful taxonomy of the cyber bullying phenomenon [54], proposed by Willard in 2006, that is centered not on instruments used but on the type of actions and behaviors perpetrated:

- **Flaming**. Use of violent messages that try to induce verbal contrasts inside forums;
- **Harassment**. Repeated sending of offensive and unpleasant messages;
- **Denigration**. To insult or defame someone online through rumors, gossip and lies, usually offensive and cruel, in order to damage the reputation of a person and his relationships;
- **Impersonation**. Identity theft: in this case, an attacker can get the personal information and data access (nickname, password, etc.) of a victim, in order to take possession of it and then hurt his reputation;
- **Outing and Trickering**. To spread the secrets of someone online, private information or personal images; push a person, through deception, to reveal confidential and embarrassing information, that can be after publicly published on the web;
- **Exclusion**. Intentionally exclude someone from an

online community (chat, forums, etc.);

- **Cyberstalking**. Repeated posting or sending intimidating messages containing threats and offenses in order to induce fear and terror to the victim;
- **Cyber bashing o happy slapping**. A criminal behavior that usually starts in real life (one or more teens harass physically a guy while someone use a smartphone to film the episode). Then the videos or images are posted over internet, usually on a social network and other users share or vote the episode;

These categories are, however, exposed to the limitations of a classification approach: the phenomena are classified according to a degree of similarity but in real life there are no clear defined limits, and some episode of cyber bullying may fall down in more than one category. So an unsupervised approach, without an a priori knowledge on classification, could be really useful in such a scenario.

4.3 Related Works

While cyber bullying is a well-studied problem from a social point of view, only recently it has attracted the attention of computer scientists, especially towards automatic detection tasks. For this reason, only relatively few articles on the subject and very few datasets are available.

Yin, et al. [55] adopted a supervised learning technique for detecting harassment, using a bag of words model based on content, sentiment and contextual features of documents to train an SVM classifier. The authors used a combined model based on sentiment and contextual features, reaching, with a support vector machine learner, a recall level of 61.9%.

Dadvar et al. [56] investigated the gender approach within the cyberbullying detection problem, applied to the social network MySpace, a platform that offers an interactive, user-submitted community of friends with personal profiles, blogs, groups, etc. Authors analyzed the content of the text written by the users but regardless of user's profile information. They used an SVM model to train a specific gender text classifier. The dataset consists of about 381.000 posts. The results obtained by the gender based approach improved the baseline by 39% in precision, 6% in recall, and 15% in F-measure.

At MIT, Dinakar et al. [57] applied different binary and multiclass classifiers on a manually labeled corpus of You Tube comments. This approach reached 66.7% of accuracy. Also, in this case authors used an SVM learner.

Kontostathis et al. [58] adopted a language based approach for cyberbullying detection. Authors collected data from Formspring.me, a "question and answer" social network, manually labelling data using Amazon's Mechanical Turk. Authors used rule based learning method and a bag-of-words approach based on a C4.5 decision tree learner and an instance-based learner. They identify true positives cyber bullying posts with an accuracy of 78.5%, while their best result obtained from a bag-of-words approach yielded to a 40% recall.

Xu, et al. [59] proposed different natural language processing techniques to identify bully traces and also defined the structure of a bully episode and possible related roles. Authors adopted Sentiment Analysis to identify roles and Latent Dirichlet Analysis to identify topics.

Cyber bullying detection is formulated as a binary (positive/negative) classification problem and a linear SVM is trained with manually labelled dataset. The results reported 89% of cross validation accuracy, showing that even basic features and common classifier, can be useful to detect cyber bullying signals in text.

We can observe that most of these studies are based on supervised approaches, and usually adopt pre-trained classifiers to solve the problem, typically based on SVM. Data are manually labelled using online services or custom applications, and are usually limited only to a small percentage. NLP techniques are obviously wide adopted in all these works, due to the strict correlation between text analysis and cyber bullying detection. Mostly NLP tasks are performed at the preprocessing stage.

4.4 Proposed Model

We want to design a new model of cyber bulling aggression, based on a hybrid set of features, starting with classical textual features but also based on the so-called "social features". Our model will avoid a bag-of-words (BoW) approach because this approach does not consider the position of words in a sentence but also because in the BoW model the feature space can be significantly large. In order to accomplish our task we manually build some features considering the cyber bullying problem from different points of view. First, aggressive sentence (bully traces) can be pre-filtered using syntactic and semantic analysis, using NLP algorithms, in order to find, for example, bad words occurrences in a document. We also consider emotional traces, inside a document that could lead to a more precise detection, introducing in the model, also sentiment analysis features. Our assumption is that cyber bullying detection can be effectively formulated as a particular sentiment analysis problem. Then, we also introduce in our model features strictly related to the social network platform addressed.

We divide features in groups, to distinguish features based on pure text analysis from features related to statistical or social analysis approach. We propose to build the model onto 4 distinct features group, divided into:

- Syntactic features (F_{syn});
- Semantic features (F_{sem});
- Sentiment features (F_{sen});
- Social features (F_{soc});

So, our global set of features F, related to a document (tweet) can be expressed as:

$$F = \{F_{syn}, F_{sem}, F_{sen}, F_{soc}\} \tag{4.1}$$

For each group we selected some features considered both from experience and from recent literature as partial good indicators of a cyber bullying sentence. For each document (tweet), we associate an input vector (2) as the weighted concatenation of the selected features:

$$x(t) = \alpha F_{syn} \oplus \beta F_{sem} \oplus \gamma F_{sen} \oplus \delta F_{soc} \tag{4.2}$$

where t is a sentence, F is a set of features, and α, β, γ and δ are weights related to each single group, that can be used to tune the model in particular context, i.e. these weights, correctly updated, could be used to apply the model to different social network platforms. In out tests, for the Twitter platform, these weights are all set to values > 1.

4.4.1 Syntactic features (F_{syn})

These features are generally obtained by statistical analysis of documents (tweets):

4.4.1.1 Bad words

From literature is quite evident and intuitive that some "bad" words make a text a suitable candidate to be labeled as a possible cyber bullying sentence. As just done in other works, we have identified a list of insults and swear words (550 terms), collecting these terms from different online available sources.

4.4.1.2 Bad words density

In our model we check also the density of "bad" words as a single feature. This features is equivalent to the number of bad words that appear in a tweet, for each severity level, divided by the words in the tweet.

4.4.1.3 Badness of a sentence

We also add a feature to our model in order to measure the overall "badness" of a tweet. This feature is computed by taking a weighted average of the "bad" words (weighted by a severity assigned).

4.4.1.4 Density of upper case letters

This feature is based on Dadvar et al. [56] results. The presence of capital letters in a tweet message is selected as a feature, considering it as possible 'shouting' at someone behavior, as commonly treated in social networks netiquette. This feature is given by the ratio between the number of upper case letter and the length (number of chars) of the whole sentence.

4.4.1.5 Exclamations and questions marks

Just like capital letters, also exclamation points and question marks can be considered as emotional comments. We just stated that cyber bullying is related to an extreme case of sentiment analysis and so it can be connected to the strong (usually bad) emotions. With this premise, we consider helpful to introduce the number of exclamation points and question marks as a feature in our model.

4.4.2 Semantic features (F_{sem})

The semantic features adopted here are based on semantic meanings, e.g. "a person", that could represent some entities extracted from documents. The assumption behind introducing semantic features in our model is that some entities tend to be more correlated with cyber bullying and, more in general, with positive or negative sentiment. These correlations can help discovering similar structures and can increase the overall accuracy of bully traces detection. For the Twitter case, we use the part-of-speech tagger developed by Carnegie Mellon [1], to detect bigrams and trigrams structures in sentences.

4.4.2.1 Bigrams

Use of offensive words is a common way of harassing someone over the web. Trivially, the adoption of foul language may be considered a sign of a potential cyber bullying episode. It's also common that when people are harassing others, they commonly tend to use personal pronouns. Therefore a good indicator of harassment can be considered the usage of personal pronouns appearing

[1] http://www.cs.cmu.edu/ ark/TweetNLP/

near bad words. From literature has been also observed that second person pronouns, such as "*you*" and "*yourself*", are more relevant respect to other possible pronouns. Using Part Of Speech analysis, it's possible to detect, as a feature, the presence of commonly occurring bigram pairs in a bullying sentence such as "*you are*", "*yourself*", and so on.

4.4.2.2 Trigrams

A still open problem in text analysis, is the negation handling. A negation (such as "no" and "not") is near to a word which precedes it or follows it, i.e. "I do not like you". The adoption of N-Gram windows inside text can help at least to mitigate some controversial sentences that contain negations, so in our model we try to detect such structures using trigrams, in order to improve the accuracy of clustering.

4.4.3 Sentiment features (F_{sen})

Sentiment analysis and cyber bullying detection were topics strictly correlated. In a cyber bullying post there is a wide range of emotions that can be used both to identify victims and bullies. Twitter posts usually contain noisy text, i.e. text that does not follow the standard rules of orthography, syntax and semantics. For polarity evaluation of a tweet we filter out words shorter than k characters, where k is an integer experimentally fixed to $k = 3$.

4.4.3.1 Sentiment polarity of a sentence

The polarity score of a single tweet is computed as the average of the sum of the polarity of its words [60]. Given a tweet message m as a collection of n words $w_1, w_2, ..., w_n$, its polarity score is defined as the mean of polarity scores of all the terms. The polarity function is calculated by using the SentiWordNet[2] lexicon.

4.4.3.2 Emoticons

Emotional signals are any information that could be correlated with sentiment polarity of a sentence. Recently in social media, users adopt visual cues that are strongly associated with their emotional states. These cues, known as emoticons (or facial expressions), are widely used to show the emotion that a user's post represents. In order to integrate these features in our model, we have built a weighted list of common emoticons (including the recent emoji list), and for each tweet we calculate the average polarity of these emoticons, if any, in the sentence. Our list comprises about 300 emoticons and *emojis* with a sentiment level associated among the values: extremely negative, negative, neutral, positive, extremely-positive. Emoticon based polarity of a tweet is computed as the mean of polarity scores of all the emoticons found in the message.

4.4.4 Social features (F_{soc})

These features are related to social behavior and their peculiarities are strictly related to the social platform analyzed. Using only features extracted from a post itself to detect harassment could be insufficient. Sometimes it's necessary to look at the context of a post to have a better understanding of the meaning and to successfully classify the tweet as a bully trace. Users who express a strong opinions on a certain topic, tend to use extreme

[2] `http://sentiwordnet.isti.cnr.it`

words in terms of emotional signals. Often, users who are familiar with each other tend to communicate in a very informal way also adopting bad words or some terms that can appear to be harassing. So it's really important to take care about the general social behavior of a user (victim or bully), to better detect eventual bully traces in posts, avoiding false positive.

4.4.4.1 Direct User Tagging

A cyber bullying sentence typically is a direct harassment to a user, especially on a social network platform. On Twitter this can be easily detected by looking at the presence of the special @USER tag. In our model this feature is added to check if a tweet contains a direct addressing to someone.

4.4.4.2 Author profiling

This feature measures the politeness of the author of posts. As previously stated, some users, mainly teenagers, adopt as a standard way of communication the use of bad words and apparently offending slang utterances. Our model tries to reflect this behavior to avoid misleading posts.

4.4.4.3 Messages exchanged with a user

This feature tries to gain information about an eventually pre-existent discussion to which the current post analyzed belongs. In these cases having an insight into the "history" of the active talk can be useful to determine if the post is a trace of cyber bullying.

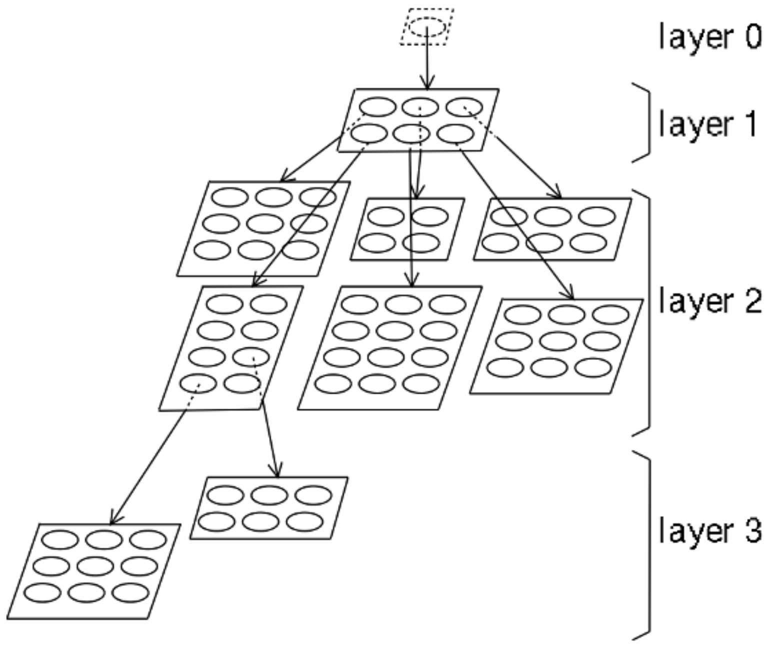


Figure 4.1: Architecture of a trained GHSOM[2].

4.5 An Unsupervised Approach

A supervised approach, with manual labelling of dataset, in case of social networks data analysis, potentially leads to a time consuming effort that could be unfeasible in certain scenarios. If we want to detect cyber bullying traces in a huge stream of data, as the ones produced by Twitter, an unsupervised approach can be more effective. The self-organizing map (SOM), also known as Kohonen map [61], is one of the most representative artificial neural network models compliant with the unsupervised learning paradigm.

As a result of the training process, similar input data are mapped in neighboring regions of the map. In the case of sentences classification, similar texts (with similar features) are grouped together. The general idea is then to display similar tweets in similar region of the map. The model consists of a number of neural processing elements, i.e. units. Each of the units i is assigned an n-dimensional weight vector m_i. Note that the weight vectors have the same dimensionality as the input patterns. The training process of self-organizing maps may be described in terms of input pattern presentation and weight vector adaptation. Each training iteration t starts with the random selection of one input pattern $x(t)$. This input pattern is presented to the self-organizing map and each unit determines its activation. Usually, the Euclidean distance between weight vector and input pattern is used to calculate a unit's activation. The unit with the lowest activation is referred to as the winner c of the training iteration:

$$m_c(t) = min_i||x(t) - m_i(t)|| \tag{4.3}$$

Finally, the weight vector of the winner as well as the weight vectors of selected units in the vicinity of the winner are adapted. This adaptation is implemented as a gradual reduction of the component-wise difference between input pattern and weight vector:

$$m_i(t+1) = m_i(t)\cdot \alpha(t)\cdot h_ci(t)\cdot [x(t) - m_i(t)] \tag{4.4}$$

4.6 Growing Hierarchical Self Organizing Map

One of the short comings of the SOM is its fixed architecture that must be initially defined. Dynamically growing variants of the SOM tend instead to produce big maps that are hard to manage. This has led to the development of the GHSOM, proposed by [2][62], which is a neural network, with a dynamic architecture, able to

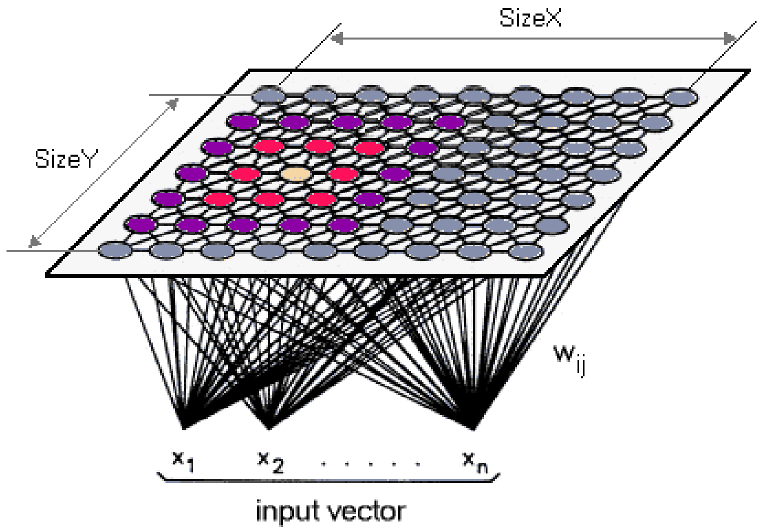
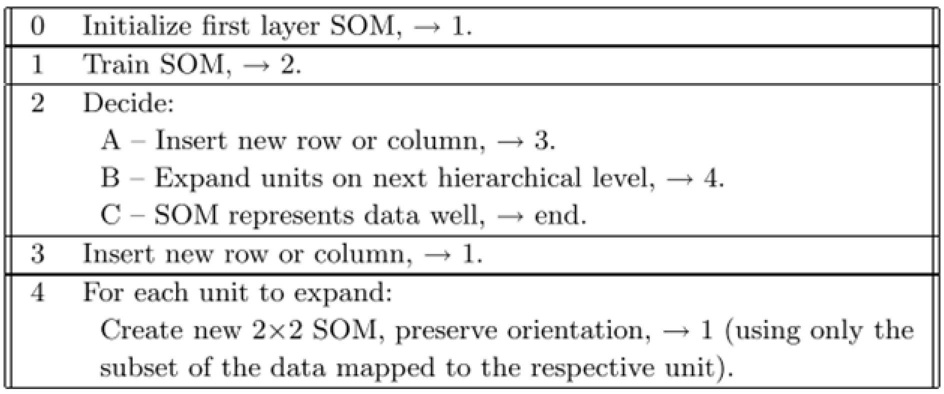


Figure 4.2: A Self Organizing Map with input N dimensional pattern.

grow in a hierarchical way according to the data distribution, allowing a hierarchical decomposition and adapting itself to the requirements of the input space. GHSOM networks are well suited in the case of large collection of documents that needs to be classified, as it happens in the case of social networks data.

The basic principle of the GHSOM is to use a hierarchical structure of multiple layers, where each layer consists of a number of independent SOMs. A single SOM is used at the root layer. For every unit in this map a SOM might be added to the next layer of the hierarchy. This principle is repeated with the third and any further layers of the GHSOM. An example of a GHSOM architecture with 3 layers is depicted in Figure 4.1. In figure 4.3 it's possible to observe an outline of the basic GHSOM algorithm. The training process of a GHSOM starts with a small map of 2x2 units at the first layer which is organized according to the standard SOM training algorithm [61]. Each single map is then expanded in order to repre-



0	Initialize first layer SOM, → 1.
1	Train SOM, → 2.
2	Decide: A – Insert new row or column, → 3. B – Expand units on next hierarchical level, → 4. C – SOM represents data well, → end.
3	Insert new row or column, → 1.
4	For each unit to expand: Create new 2×2 SOM, preserve orientation, → 1 (using only the subset of the data mapped to the respective unit).

Figure 4.3: Outline of the GHSOM algorithm [2].

sent the corresponding subset of data at the specific level of granularity. After the model vectors have converged some decisions are made. Namely, $2A$ if a new row or column should be inserted to improve the quality of the representation on this level, $2B$ if the quality of the representation should be improved on the next hierarchical level by expanding one or more units, or $2C$ if the SOM represents the data well in its current form. These decisions are the core of the GHSOM algorithm as they control the hierarchical structure and the shapes of the individual SOMs. In case of $2A$ a new row or column is inserted and the training process is repeated. In case of 2B for each map unit which needs to be expanded a new 2x2 SOM is created in such a way that the overall orientation is preserved. The training process is repeated for each of these new maps. Otherwise, $2C$ if neither decisions $2A$ nor $2B$ are made no additional steps are necessary.

4.7 Methodology and results

Automatic solutions related to cyber bullying detection are not properly studied in the past. This is one of the main reason for which there exists insufficient training datasets available. Some datasets are available instead on general sentiment analysis and all of them are used in supervised approaches. Although bullying messages are posted every day compared to hundreds of thousands of messages posted every second, they are very sparse. Collecting enough training data is an actual big challenge since random sampling will lead only to few bully messages. We selected two distinct dataset, recently published, related to the social network FormSpring.me [58] and one related to YouTube [56].

4.8 Conclusions

In this chapter we proposed to adopt an unsupervised approach to detect cyber bully traces over social networks, based on Growing Hierarchical Self Organizing Map. Our new designed model comprises several hand crafted features that are used to catch semantic and syntactic communication behavior of potential cyber bullies. We conducted some experiments on datasets taken from literature, like those coming from FormSpring, and YouTube, but also on a real data stream, collected from Twitter. Results, reported in chapter VI, indicate that our model achieve reasonable performance and could be usefully applied to build concrete monitoring applications to mitigate the heavy social problem of cyber bullying. Indeed, there is plenty of room for improvement on these techniques (i.e. sarcasm identification) in order to achieve better results, as will be discussed in chapter VII.

Chapter 5

Applied Natural Language Processing techniques to data

"I was terrible in English.
I couldn't stand the subject.
It seemed to me ridiculous to worry about
whether you spelled something wrong or not,
because English spelling is just a human
convention - it has nothing to do with anything
real, anything from nature."

- Richard P. Feynman

In this chapter we will describe some basic techniques in the field of NLP (Natural Language Processing) that we adopted in this thesis, in order to filter and clean data taken from networks (i.e. movie reviews), and therefore obtaining more useful data to feed in the neural networks proposed models. The aim of this chapter is twofold. First, once the data are cleaned, we want to find an effective representation of these sentences in order both to keep semantic properties of text, but also to deal with a feasible dimensionality of features vectors. Second, we want also to adopt the Italian language in our Sentiment Analysis task, and so we need to measure (empirically) the performance of the selected algorithms and relative datasets in producing this representation.

5.1 Introduction

We saw in the second chapter that the complexity of the languages places important limits on the performances of automatic text processing tools, especially for tasks such as Sentiment Analysis. Fortunately not all the complexity of the language is necessary to realize effective text analysis tools. The first essential step is to make a quantitative representation of texts in order to treat sentences by statistical models. In this chapter we will describe some useful methods, adopted in this thesis, to get an effective representation of text sentences, in order to make it usable as input to machine learning algorithms. We will also discuss about the pre-processing text techniques which allow to filter and refine the representative models obtained.

5.2 Text pre-processing techniques

Generally, textual sentences contain many words or auxiliary symbols (punctuation) that can be filtered through a preliminary analysis. Some algorithms are more efficient on short texts while others techniques work better with longer texts but, regardless of length, all methods provide substantially a similar process that operate a reduction of text in arrays of data [63]. A typical initial procedure applied in cases of textual analysis, is the so-called pre-processing phase, generally applied to delete useless information that appear in the text. In order to reduce the dimensionally of words in documents, special methods such as filtering and stemming are applied. Filtering methods remove those words and symbols, from the whole set, which do not provide relevant information:

- ***stop word filtering*** is a standard filtering method. Words like prepositions, articles, conjunctions, etc. are removed from sentences due to the fact that these words do not contain any information. Elimination of stop words has an additional important benefit. It reduces the size of the indexing structure considerably. Words that within a corpus appear too frequently (for example in 90% of texts or more) or too rarely (less than 5% of the texts) can usually be removed without reducing information on the data structure.
- ***stemming methods*** are used to produce the root from the plural or the verbs. Frequently, users specifies a word in a sentence but only a variant of this word is present in a document. This problem can be partially overcome with the substitution of the words by their respective stems (roots). A stem is the portion of a word which is left after the removal of its affixes (i.e. prefixes and suffixes). Stems are thought to be useful for improving retrieval performance because they reduce variants of the same root word to a common concept. Furthermore, stemming has the secondary effect of reducing the size of the indexing structure because the number of distinct index terms is reduced.

If you delete information on the order of appearance of words in a text, then you go to the so-called **bag of words** model that is: a set of terms that does not take account of their order and position. The literature shows that it is indeed possible to reduce the text to a smaller set of terms (called stems), still getting a valid representation of the text.

Stems made of a single word are called **unigrams**, while stems made of a couple of words are called **bigrams**, and so on with **trigrams** and **n-grams**. In general, from literature we know that adopting stems with more than three words does not provide any special additional information and does not increase the quality of the classification. Therefore stemming procedures most used are limited to unigrams. It's important to underline that this pre-processing phase must be designed and optimized for each single language.

5.3 Corpus creation in Italian laguage

Most of the research activities on Sentiment Analysis available in the literature are all focused on the English language, and in the same way most of resources necessary for NLP tasks, such as lexicons and corpora, exist almost exclusively in English. This lack of resources based on a language different is thus a critical first issue to be addressed for any research in the field of text analysis, which is not based on English. In the case of this thesis, we want to adopt the Italian as the reference language content. From an initial analysis of any resources currently available in Italian, it was possible to trace only 3 existing corpora, which are here listed below:

1. "La Repubblica" Corpus
2. itWaC (University of Bologna)
3. Paisá (CNR - Consiglio Nazionale delle Ricerche)

These corpora will be analyzed and compared, in the next paragraphs, in order to select the one best suited to our thesis aims.

5.3.1 "La Repubblica Corpus"

The "La Repubblica Corpus" [64] is a large corpus based on collection of texts taken from Italian newspapers (with about 380 million tokens), annotated with Part of Speech (POS) tagger developed and published by the University of Bologna (SSLMIT[1]). Although not intended as corpus reference, this corpus, which collects the annals of the newspaper "La Repubblica" from 1985 to 2000, is characterized by its large size (about 325 millions of words) and the query interface (freely accessible after registration) which allows advanced searches that use, among other things, meta-data, parts of speech and entries. The corpus format is: token, POS-tag (created with the TreeTagger[2] tool trained up with ad hoc resources), lemmatized (with the open source software tool Morph-it[3]) and classified with an approach based on an SVM trained with ad hoc resources. The labels provided by the classifier are related to different topics such as: religion, culture, economy, education, news, politics, science, society, sport, weather, etc. The single articles in the corpus are structured into the following text components: title, subtitle, summary, text, information and meta-data such as author and publication year.

5.3.2 "itWaC"

itWaC is the larger corpus currently available in Italian language [65]. It is made of texts downloaded from the web with automated methods. It contains more than a billion and a half of words. The corpus can be freely downloaded after registration and users can have an on-

[1]http://www.sslmit.unibo.it
[2]http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/
[3]http://dev.sslmit.unibo.it/linguistics/morph-it.php

line payment service that offers a lot of advanced consultation interface. This interface allows, among other things, to extract the typical object complements of a verb, to compare the most characteristic adjectives of a certain noun with respect to another, etc. The itWaC corpus reflects the recent trend to build corpora collecting web texts with automatic procedures. Actually there are many tools that allow users to build their own corpus from the web using similar procedures. itWaC is obtained in part also with using *medium-frequency* words taken from the corpus of "La Repubblica", of which this corpus represents substantially an extension. itWaC has been tagged (POS) with TreeTagger and also lemmatized using the Morph-it tool.

5.3.3 "Paisá - CNR"

The Paisá[4] corpus is a large collection of texts downloaded from the web, in Italian language (with about 380.000 documents and about 250 millions of tokens) and protected by Creative Commons licenses. The documents in the corpus Paisá were selected using two criteria. The first criterion, inspired by the WACKY project[5], is that we should identify the URL (Uniform Resource Locator) of the documents to be downloaded by searching for random combinations of words on the Yahoo search engine. For the Paisá corpus the words used were taken from the basic vocabulary of the Italian language Paravia, organized into a list of 50,000 pairs. The research of documents was limited to pages in Italian language, always with Creative Commons license guaran-

[4]The Paisá corpus is made available at http://www.corpusitaliano.it through a Creative Commons license. The rights of web texts remain with the owners of the URLs who have in turn made them available through Creative Commons licenses.

[5]http://wacky.sslmit.unibo.it/doku.php

teed. Once the list of URLs has been completed, the building process of the corpus proceeded with the elimination of pages mistakenly attributed to the CC licenses, identified on the basis of a blacklist of sites manually realized during the implementation of previous and experimental versions of the corpus, as well as downloading and cleaning up the documents with the KrdWrd software system[6].

A second criterion of Paisá building strategy includes documents originating from the Italian versions of some of the Wikimedia Foundation, such as *Wikipedia, Wikinews, Wiktionary, Wikibooks, Wikiversity, Wikivoyage.* In this case, they have used the official dumps issued by the Wikimedia Foundation, extracting the text with Wikipedia Extractor tool. Once obtained all the materials, it is carried out a skimming the entire collection in order to eliminate voids documents or with lower amounts of text to 150 words. The corpus contains a total of about 380.000 documents from about 1.000 separate sites, for a total of about 250 millions of words. About 260.000 documents come from Wikipedia, about 5.600 from other Wikimedia Foundation projects. About 9.300 documents come from Indymedia web site, and it is estimated that the remaining about 65.000 documents come from blogs. The data of the corpus has been collected during the period that goes from September 2010 to October 2010.

The building steps of the Paisá corpus are below reported:

1. Creating a seed list of random combinations of frequent Italian words;
2. Finding the URL address through the Yahoo search engine and cleaning the list of URLs;

[6]https://krdwrd.org/

3. Download the contents of the URL addresses and creation of cleaned corpora;
4. Perform linguistic annotation on data;
5. Meta data creation;
6. Indexing, using CWB (Open Corpus Workbench) tools[7];

The documents are enclosed in the body of an XML element *text* with an *id* attribute (a distinct numerical value assigned to each document) and URL, which contains the address from which the document was downloaded.

5.4 Corpus selection criteria

The choice of the corpus to be used in this thesis falls on the corpus Paisá produced by the Italian CNR, for many reasons. The nature of the present work, applied mostly to web users and social networks, suggested to adopt a linguistic context closer to documents retrieved from Internet. The selected corpus of the CNR, being composed mainly by documents taken from the web, responds better to our requirements, i.e. analyze posts and comments over social networks, respect to the other corpora listed like ”itWaC” and ”La Repubblica”, that are mainly collections of news articles. Furthermore the Paisá corpus results is more updated than its competitors, in terms of year of creation, i.e. 2010 vs 2004. In the next paragraph we will explain how this corpus will be used to generate vectorized representation of text using the Word2Vec algorithm.

[7]http://cwb.sourceforge.net

5.5 The Word2Vec algorithm

Word2Vec[4] has been recently introduced by Mikolov (2013) and it immediately have drawn, as part of the Natural Language Processing, a great deal of attention in last years. Vectorized representations of words, learned through the Word2Vec algorithm, have proven to efficiently manage semantic meanings of those words, which become very useful in the field of NLP applied to Sentiment Analysis. More in detail Word2Vec is an efficient predictive model capable of learn words representation from unstructured text. This technique used in Natural Language Processing for text analysis is based initially on the training of a *shallow* neural network, on a corpus V of sentences, in order to associate to a single word $w \in V$ a vector of real numbers, with a certain dimensionality choice. The Word2Vec technique allows to automatically define a metric representation in the space of words, that is able to represent semantically similar words with neighboring points in same the vector space. The Word2Vec model is available in two different versions, a model based on **Continuous Bag of Words** (CBOW) and a model based on **Skip-Gram**. Algorithmically, these two models are very similar, except for the fact that the CBOW version of the algorithm predicts a target keyword from words of context (neighboring words), while the skip-gram version makes the reverse, and it provides the context or the neighboring words, given a target word. From a statistical point of view the CBOW approach appears to perform best on small datasets, while the skip-gram approach has been proved to be more efficient for big dataset, as in the case of the corpus adopted in this thesis, in made of about

1.3 millions of documents. In the figure 5.1 it is possible to schematically show the two different models of Word2Vec, where $w(i)$ indicates the *i-th* word of the analyzed sentence.

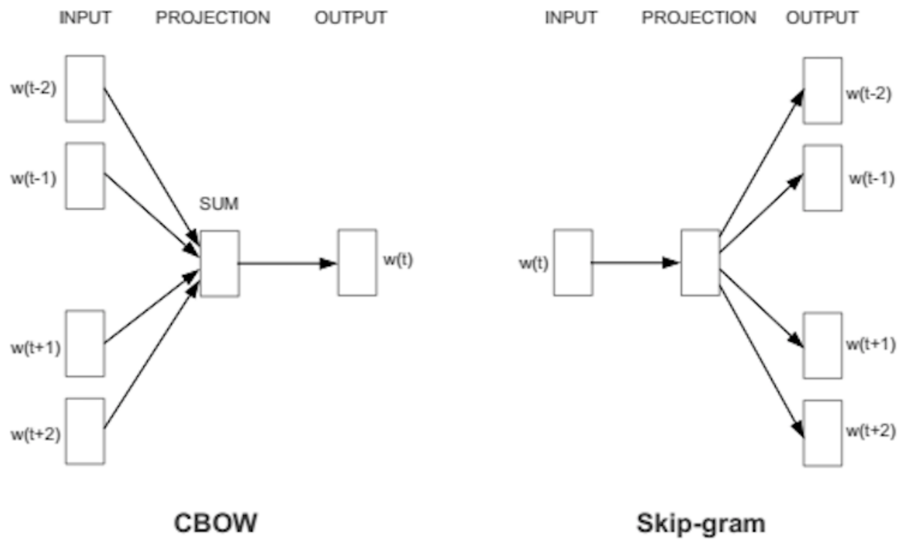


Figure 5.1: Comparing CBOW model and SKIP-GRAM for word vector.

Before using Word2Vec out of a corpus we applied some pre-processing operations, such as the deletion of the marks and the removal of some very common words known as ”stop words”. For our purpose we drawn up a list of 400 common words used in Italian language. The list has been compiled from sources already available over the web. An interesting feature that this algorithm produces over the representations is the fact that it allows to perform also some mathematical operations on words: for example, taking three words like ”king”, ”man”, and ”woman”, using a simple algebraic operation like: ”ing” - ”man” + ”woman” you can get as a result the word ”Queen”. Another useful feature is the ability to categorize (create clusters) of similar words to each other: for example, searching for the word ”cat” you will have a subset of the words ["cat", "cow", "dog"] that is possible to label with the class "animal". These operations are possible because, as previously said, each word is associated to a vector that allows to map each word in a vector space, therefore, similar words will have vectors similar to each other and will therefore topologically be near in a hypothetical grid, as can be seen in the figure 5.2, where a subset of common adjectives in Italian language are mapped on the vector space provided by the Word2Vec algorithm, using a t-Distributed Stochastic Neighbor Embedding [8](t-SNE) graph.

Also the representation generated by Word2Vec is able to define topological relationships between pairs of similar words. For example, in the case of the word "man", this term has an implicit relationship with the keyword "woman", and the same relation keeps valid if we use as word pairs the terms "king" and Queen". Similarly, as another example, there is a topological relationship between words that indicate a capital and a nation, i.e. the words "Italy" and "Rome". Some of these relationships are showed in figure 5.3

With this topological proximity principle, also many other types of relationships can be established, such as between verbs and tenses. Finally, a very interesting aspect is that vectors generated by the Word2Vec algorithm are usable and easily understandable by deep neural networks. In this way, the training stage of the networks became simpler being the obtained vectors already a good initial representation of the input, that represents a fundamental requirement for a machine learning based approach.

[8]t-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. The technique can be implemented via Barnes-Hut approximations, allowing it to be applied on large real-world datasets.

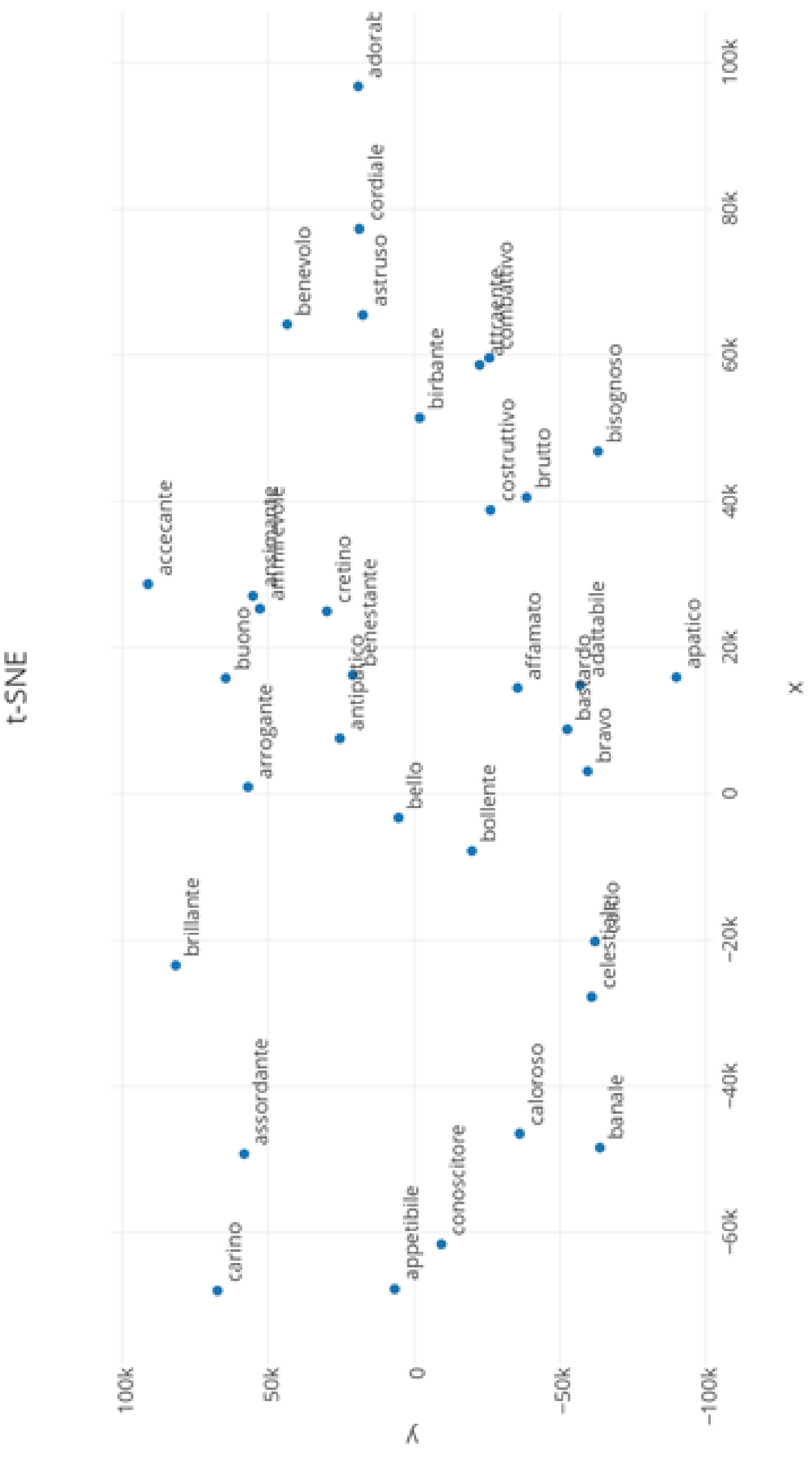


Figure 5.2: Word2Vec words diagram for Italian language.

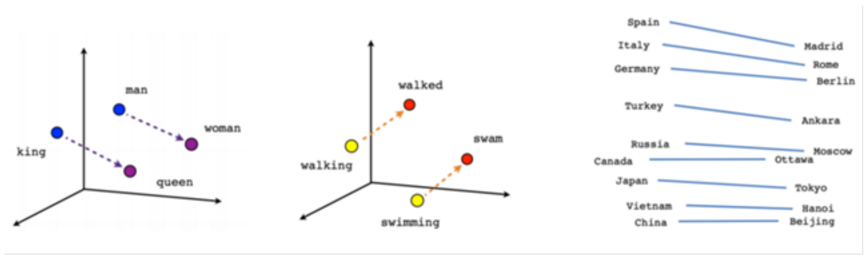


Figure 5.3: Examples of topological relationships between related terms in Word2Vec.

5.6 Conclusions

The training Word2Vec model applied on the selected Paisá corpus has led to obtain a vector representation of about 247.000 words in Italian language. Several tests have been performed in order to select the best vector dimension (i.e. the number of features) to be used for the representation. Our choice, for both qualitative and performance reasons, falls on a number of features equal to 300, as a good trade-off experimentally found. Values above this number lead to important delays of the training phase, in terms of tens of hours, whereas values below this threshold does not allow to obtain a good representation of the vectors in terms of similar words.

In table 5.1 it's possible to observe the quality of the obtained representation, comparing the results with two different sampling (1M and 3.2M of sentences) of the used corpus.

We can observe the expected improvement, from a qualitative point of view (i.e. the set of similar words semantically closed), compared to a group of random tokens (*buono, martedí, fantastico, morte, vita)*, respect to 2 different sizes of the corpus used (Paisá) in Italian language. It's easy to observe that the last column of table 5.1 contains more terms semantically (about 90%)

Table 5.1: Comparison results obtained applying the Word2Vec algorithm to different corpus size (Italian words).

Token	**Similar 10 words on 1M sentences**	**Similar 10 words on 3.2M sentences**
buono	niente, degno, cattivo, brutto, veramente, caro, giusto, bello, colui, buon	stupido, cattivo, brutto, giusto, bello, gentile, ingenuo, generoso, bene, onesto
martedi	lunedi, venerdi, mercoledi, estoniaitalia, giovedi, sabato, festivi, domenica, patronale, pomeriggio	lunedi, mercoledi, venerdi, giovedi, sabato, domenica, pomeriggio, pomeridiana, mattina, domeniche
fantastico	fantascientifica, ambientato, fantascienza, supereroico, immaginario, akutagawa, disneyano, cthulhu, fantascientifico, puh	fantasy, lovecraftiano, ambientazione, fiabesco, fantascienza, bildungsroman, fantasia, immaginario, narrativo, fantastica
morte	pulcheria, avvenuta, onoria, morto, valentiniano, tenji, esilio, eracleona, figlio, mori	avvenuta, prematura, agonia, morto, morendo, morirá, decesso, assassinio, scomparsa, mori
vita	spirituale, therese, crizia, esperienza, visse, vissuto, visione, storia, vivere, realtá	solitudine, vissuta, dissoluta, consapevolezza, ascetica, quotidiana, vecchiaia, vivere, terrena, condizione

related to the reference token, than the second column, which shows correct related terms only in the about 60% of cases.

Chapter 6

Dataset and results

"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong."

- Richard P. Feynman

In this chapter the results of our experiments with the proposed neural network models will be shown, for both the general Sentiment Analysis task and the automatic detection of cyber bullying traces.

6.1 Introduction

This chapter is divided in two sections. In the first section, relative to the general Sentiment Analysis problem, we will describe in details our data sets and the obtained results, using the Deep Belief network model proposed in chapter III.

In the second section we will report the results obtained with the proposed unsupervised approach model built on a set of hand crafted features, discussed in chapter IV, for the automatic detection of cyber bullying traces, using a Growing Hierarchical Self Organizing Map (GHSOM).

6.2 Sentiment Analysis with Deep Learning

We have discussed in chapter III our proposed solution to the Sentiment Analysis task, done at a sentence level, and focused on Italian language. The model, that here we will resume shortly, is based on a semi supervised approach and adopts a Deep Belief Network structure. In order to achieve an efficient representation of the text used as input to the neural network, it was necessary to study and subsequently adopt an algorithm able to convert sentences into numerical values (in our case vectors of real numbers), that keep also the semantic properties of the text, required for the proper implementation of a Sentiment Analysis task. We have choosen to adopt the Word2Vec algorithm, recently proposed by Mikolov in 2013, and illustrated in chapter V.

One of the objectives of our research was to take the Italian language as the reference language, and this has made unfeasible to use exiting corpus and existing vector representations (in English), and so it was therefore necessary to select a representative corpus (Paisá) that was also compatible with the context of study (social networks).

The neural network model designed for Sentiment Analysis activities was based on a Deep Belief network structure, obtained "stacking" multiple Restricted Boltzmann Machines networks. We have tested different structural models of this deep network, in terms of the shape of the network, or by varying the size of input level or by varying the number of hidden levels and related neurons.

6.2.1 Experimental Setup

Some established choices of our model are represented by a SoftMax classifier (on the output top level of the network), with an error function based on the Negative Log Likelihood. The type of neurons used as input to the network is of the Gaussian type. Our choice, for the activation function of the hidden layer nodes, is ReLu, that by testing tends to create more robust activations and improves the evaluation of F1 score. The initialization of the input level weights was made through the XAVIER algorithm, proposed by Xavier Glorot and Yoshua Bengio (2010).

Note that the size of the input layer corresponds to the size of the vectors (k) generated by the Word2Vec algorithm. The generation of the data input to the neural network was carried out, starting from a single sentence of variable length, by first applying a transformation vector of each token (*word*) of the sentence and subsequently by performing a mean (*average*) of the vectors of the individual token to order to obtain a single vector representative of the sentence. For the hyper-parameters configuration we performed several tests reported in chapter III, that we will summarize in this chapter.

A first class of experiment was performed by varying the number of hidden neurons of the model. We saw earlier in Chapter III that there is a specific and defined theory that suggests what the proper structure of a network Deep Belief should be. From the comparative study of Larochelle (2009), we know that using the same size for all levels generally works better, or the same way of using descending size (pyramid style) or increasing (inverted pyramid), but we also know that the choice is definitely dependent on the data used.

6.2.2 Datasets

In order to train the Deep Belief Network two different datasets have been used, both in Italian language and both relative to movie reviews, publicly available on the web.

The first dataset used has been built upon the website `www.cinemadelsilenzio.it` and it comprises about 7.000 reviews written both by experts and registered movie fans users. Each single review expresses a numeric evaluation (from 0 to 10) of a film, with also intermediate values (e.g. 7,5).

Considering the average of evaluations, we have labeled reviews rated from 0 to 6 (included) as negative and reviews rated with a vote greater than 6 till 10 as positive. It is worth to note that this dataset, also due to the fact that many opinions are written by cinema experts, contains reviews expressed in technical language that can be considered a real film criticism. We report here a sample of such content (both in Italian and English language):

> *Il film é un buon esempio di quando il cinema italiano spalanca le sue finestre lasciando entrare una vivace brezza che lo rende piú internazionale e fruibile anche fuori dai propri confini. Sebbene la sceneggiatura si lasci cullare in qualche appetibile cliché commerciale, lo stile, il ritmo e l'interpretazione coinvolgenti ci fanno chiudere un occhio volentieri e sognare.* (Votazione 7,5). (ITA)
>
> *The film is a good example of when Italian*

cinema opens its windows letting in a brisk breeze that makes it more international and usable even outside its borders. Although the script will let yourself in some attractive commercial cliché, the style, the rhythm and the engaging interpretation make us turn a blind eye willingly and dream. (Rated 7,5). (ENG)

These kind of textual expressions, inside the dataset, that include metaphors, actually makes the realization of a Sentiment Analysis task even much more complex and challenging. Finally, the dataset results to be quite well balanced (45% negative classes and 55% positive classes).

The second dataset used, for the training stage of the Deep Belief Network, is still made of about 10.000 movie reviews, taken from the website `http://www.filmup.com`, and it comprises reviews written by common users (not experts). The votes in this case are expressed with integer values ranged from 1 to 10, without intermediate evaluation. The reviews are made of a title and the real textual review. An example of a single review taken from this dataset is:

Titolo: *Banalissimo e per nulla magnifico.*
Recensione: *Mi scuso per il titolo, ma a me questo film non é per nulla piaciuto!* (Votazione 4). (ITA)

Title: *Banal and not at all magnificent.*
Review: *I apologize for the title, but I don't like this film at all!*

(Rated 4). (ENG)

It was possible to empirically observe that the reviews of this data set are much more direct and synthetic, probably also due to the fact that have been inserted by a younger audience. The title, in most cases, is a good example of clear and immediate opinion. In contrast, this dataset contains many expressions in "slang" idioms that complicate the analysis.

6.2.3 Results

This section will summarize and discuss the test results numerically reported at the end of chapter III (tables 3.1, 3.2). Experimentally, once fixed the dataset size, we evaluated the response of the network in terms of classification *accuracy*. In order to also validate the correctness of the representation of the input vectors dimension some comparisons were made, that are below reported using different charts. As an example, a general axis label $S1-Si-Sn$ indicates that $S1$ is the size of the input vector (visible layer), Si is the dimension of an intermediate hidden layer, while Sn is the size of output level (the final classifier). The tests were carried out both on the data set A (*"Cinema del Silenzio"*) and on the dataset B (*"Filmup"*). In the next charts below we report, as an example, how the accuracy response on the network can change respect to different structure (levels) configuration, and on a basis of 1000, 2000 and 3000 batch size of the data set (in this case A).

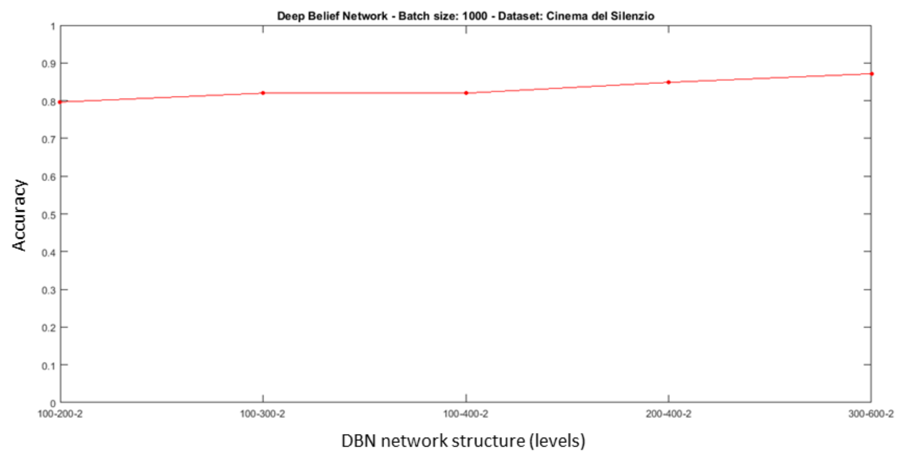


Figure 6.1: Response of DBN in terms of accuracy varying levels size, on dataset (A) with batch size 1000.

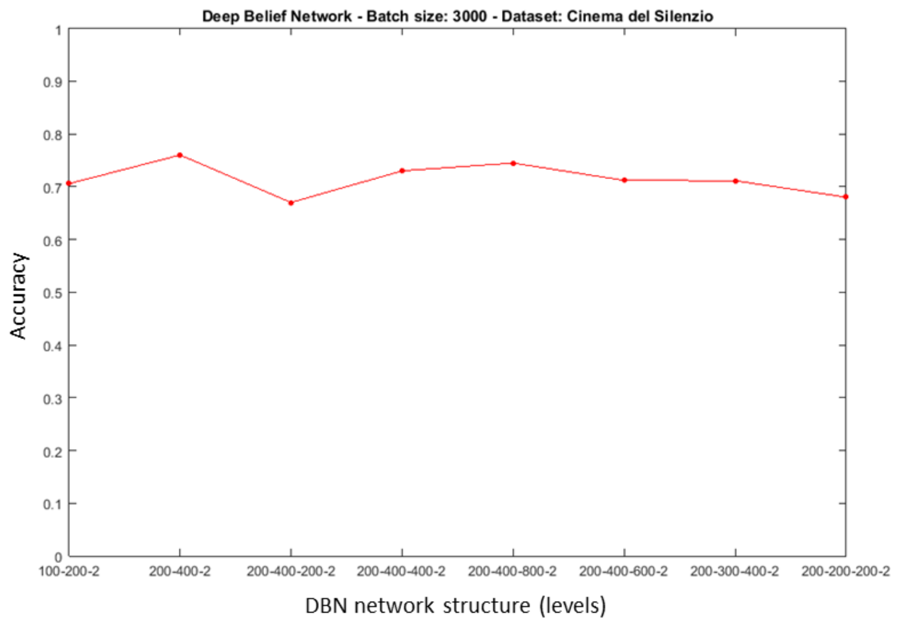


Figure 6.2: Response of DBN in terms of accuracy varying levels size, on dataset (A) with batch size 2000.

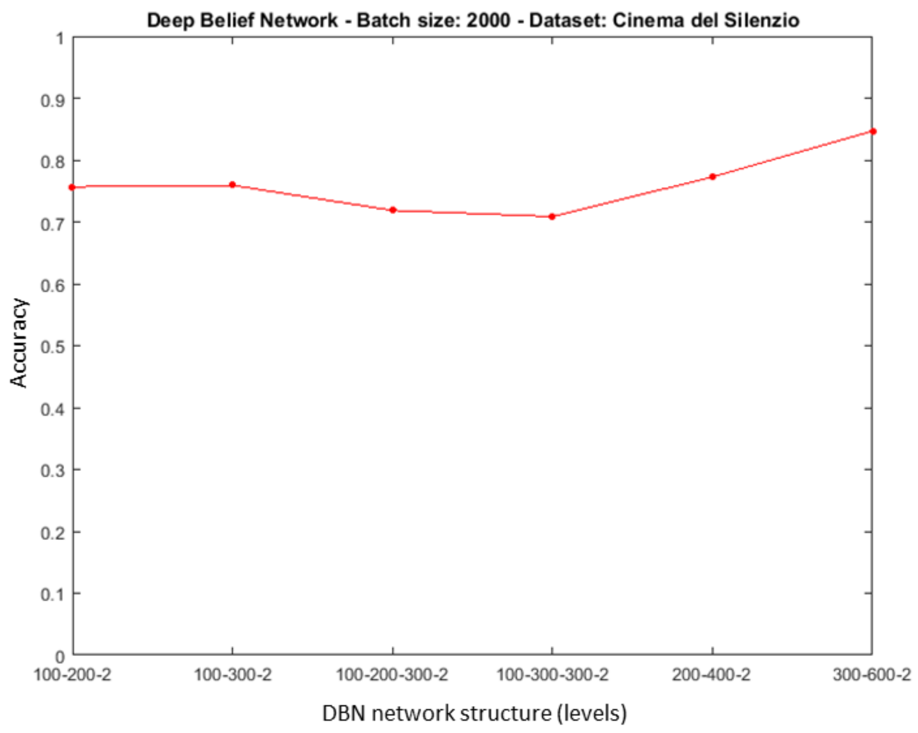


Figure 6.3: Response of DBN in terms of accuracy varying levels size, on dataset (A) with batch size 3000.

The dataset (B), taken from the web site `http://www.filmup.com` allows to have a larger number of available training data. Some tests have been performed in order to understand how was the network response respect to the increasing size of the data set, keeping fixed the structure of the DBN network. In the next chart 6.4 it's possible to observe the accuracy of a 300-600-2 structured based network, compared with increasing size (batch) of the data set (B).

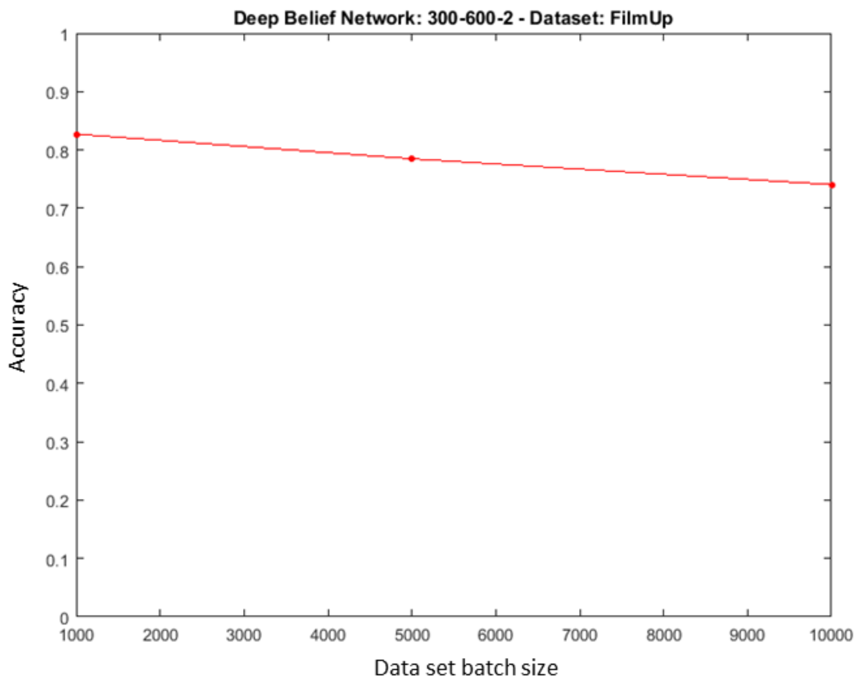


Figure 6.4: Response of DBN in terms of accuracy varying the data size, on dataset (B) with structure 300-600-2.

The possibility of having a large data set also allowed to perform tests on more complex configurations of DBN networks in terms of the size of the hidden input, as showed in the next chart in fig. 6.5.

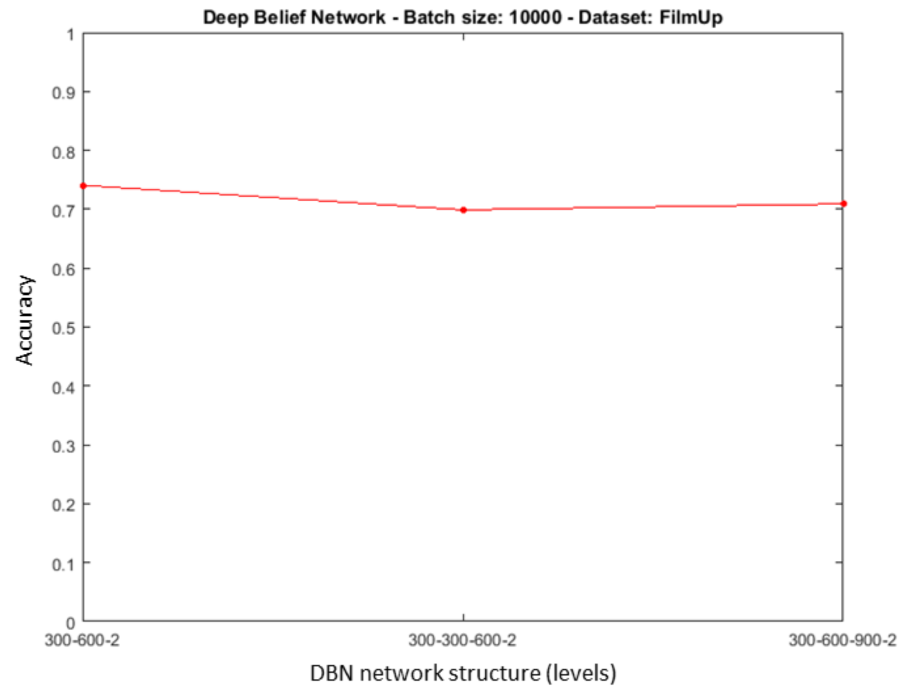


Figure 6.5: Response of DBN (accuracy) on three network configurations, on dataset (B) with batch size = 10.000.

In line with what occurred in the literature, from the tests that were carried out it was found that a first hidden layer, configured so as to be larger (i.e. doubled) than the dimensionality of the input data, actually works better than one with a reduced dimensionality.

In the following table 6.1 we report a summary of the most significant cases of our testing activities done with the proposed model. We underline that the training dataset were all validated by K **Cross Validation**[66], with $K = 10$.

In order to evaluate the correctness of our approach it is worth now to compare these results with also different datasets and similar models taken from literature that address the same objective (i.e. Sentiment Analysis on movie reviews, applied at sentence level). To accom-

Table 6.1: Summary of the main results obtained from the experiments carried out varying the DBN configuration parameters.

Network structure	**Batch size**	**Epochs**	**Dataset**	**Accuracy**
100-200-2	500	2	A	0,88
100-200-2	1000	20	A	0,8243
100-300-2	1000	1	A	0,8229
100-400-2	1000	5	A	0,82
200-400-2	1000	30	A	0,8757
300-600-2	1000	5	A	0,8714
300-600-2	1000	5	B	**0,9171**
300-600-2	5000	30	B	0,7849
200-400-800-2	3000	5	A	0,7448
300-600-900-2	1000	10	B	0,7829

plish this evaluation we take into account the classical reference work entitled *"Thumbs up?: Sentiment classification using machine learning techniques"* published by Pang and Vaithyanathan in 2002 [18]. In this work authors adopted a widely-used movie review dataset called MOV, with 2.000 labeled reviews (1.000 positive reviews and 1.000 negative reviews). A recent model comparison, using also DBN networks, is available in the work done by Zhou et al. in [1]. In table 6.2 it's possible to see four different representative semi-supervised learning methods to which we compared our DBN model (labelled as DBN-W2V). The first model is the Spectral Clustering[67], introduced by Kamvar et al. in 2003. The TSVM model refers to the Transductive SVM[68] introduced by Collobert et al. in 2006. In the HDBN[1] model, Hybrid means that this architecture use a modified RBM layer (a convolutional layer), and the input vectors are represented using a BoW approach, and with a network layers configuration of 100-100-4-2 neurons. Our DBN-W2V model, applied on the same MOV dataset, with a network structure of 100-200-2 neurons, performs quite

Table 6.2: Comparison of our DBN with other models [1], on MOV dataset.

Network Type	**Accuracy**
Spectral	67,3
TSVM	68,7
HDBN	72,2
DBN-W2V	**76,2**

well also when compared to other models.

6.3 Cyberbullying detection with GHSOM

In this section we will summarize the results obtained from our proposed unsupervised approach, that aim to automatically detect cyberbullying trace in a social networks, as introduced in chapter IV. We treated this problem as a narrowed Sentiment Analysis task, in which we mixed, in a hand-crafted features model, both extremely negative polarity detection model and specific features related to the social network analyzed.

6.3.1 Experimental setup

Our experiments are based on a GHSOM network, with a final grid of 50 x 50 neurons, and about 20 hand-crafted features as input layer, as described above. All the input vectors have been normalized between 0 and 1. The network has been trained with a learning rate of 0.7, and with 10.000 epochs. We have implemented the GHSOM network algorithm using the SOMToolbox[1] open source framework. Some pre-processing has been applied to the dataset, as stop words removal, punctuation removal and stemming. To measure the goodness of generated clusters we used standard classification measures like *Precision* (i.e., for a given class X, how often tweets are classified as X when they should not be, or also a measure of false positives), *Recall* (i.e., for a given class X, how often tweets are not classified as X when they should be, or also a measure of false negatives) and *F-Measure*, an harmonized mean of precision and recall, also called F1 score. The results for each measure will be in range between 0 (worst) and 1 (best). The formulas for calculating these results are shown below:

[1] `www.ifs.tuwien.ac.at/dm/somtoolbox`

- $Precision = TP/TP + FP$
- $Recall = TP/TP + FN$
- $F - Measure = 2((PR)/(P + R))$

where TP = true positives, FP = false positives, TN = true negative, and FN = false negative.

In all our tests we adopted K-fold cross validation. Then the average error across all K trials has been computed. The goal of cross validation is to define a dataset to "test" the model in the training stage (i.e. the validation dataset), in order to limit problems like overfitting, and also to give an insight on how the model will generalize to an independent dataset (i.e. an unknown dataset from a real problem), etc. One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds. We have trained and tested our GHSOM neural network respect to a k-folded dataset, applying a k-fold partitioning of data, with $k = 10$. The dataset is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other $k - 1$ subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set $k - 1$ times. The variance of the resulting estimate is reduced as k increased. The disadvantage of this method is that the training algorithm has to be re-run from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set k different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over. So we divided randomly the dataset in 10 parts, and for each experiment we used 9 parts of the dataset for train and the remaining part for test, changing each time the part related to test. The starting data set can be considered highly unbalanced due to the fact that positive classes are only a small number. So we have also applied a "random under sampling technique" [69] that decreases the majority class so as to match the minority class, in order to balance the classes distribution.

6.3.2 Results on Formspring.me dataset

In our first experiment we considered the dataset from Kontostathis et al.[58], collected from the social network Formspring.me, from September 2011 to July 2012. Formspring.me is a *question-and-answer* based platform where users invite others users to ask and answer questions. Anonymity guaranteed by this platform makes it a fertile ground for episodes of cyber bullying. The data were manually labeled using Amazon Mechanical Turk, where 3 experts manually annotated about 13.000 questions and related answers, with an average of 6% of bullying posts. Totally, the dataset includes 20.921 questions and answers. We considered records having at least 3 positive annotations, i.e. records that can be labelled with a good certainty as bully traces. The dataset contains at the end 1.239 positive classes, and 19.682 negative classes. The dataset is clearly unbalanced for the

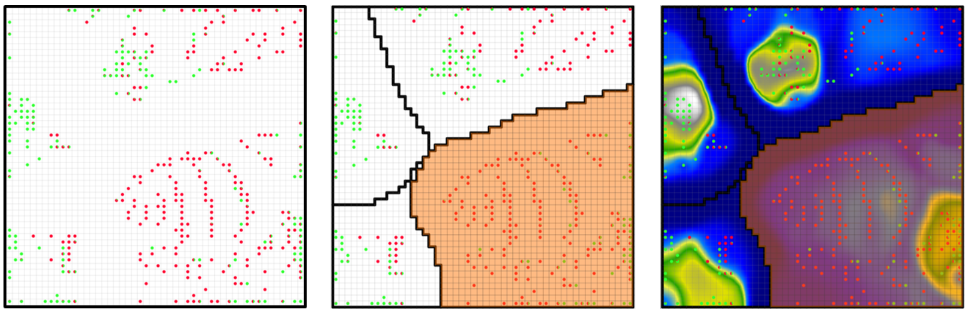


Figure 6.6: Class distribution (left) over the GHSOM with K-means clustering (center) and smoothed density map (right). Red dots mean cyber bully traces while green dots mean non bully traces (FormSpring.me dataset).

training stage, so, as stated before, we choose to subsample data adopting a random subsampling technique, and providing a final balanced dataset with about 1.200 posts (with 650 negative classes and 550 positive classes). So, at each round of our K-fold cross validation ($K = 10$) we have about 1.100 sentences used as train dataset and about 100 sentence used as test dataset.

In fig. 6.6, we show examples of clusters identified by a K-means algorithm applied to generated network lattice, during a test run. We have choosen to evaluate a single cluster (the biggest one) that can be automatically discovered. We evaluated here the results of the GHSOM, superimposing on the winners neurons the supervised classes provided by the dataset. It's possible to see in the right-most column of fig. 6.6 how the dataset is topologically mapped into clusters, and how some of these clusters seem to be quite homogeneous in terms of associated classes. We then applied clustering algorithms to see how automatically it's possible to detect these clusters in order to automatically group and then classify the input dataset.

Because of the specific interest in the accurate detection of hateful content, the results reported in Table 6.3,

Table 6.3: Results obtained on FormSpring.me dataset.

Precision	**Accuracy**	**Recall**	**F1**	**Method**
0.72	0.73	0.69	0.71	GHSOM
0.60	-	0.40	-	C4.5
-	-	0.67	-	SVM

are related to the positive class only.

Finally these results can be compared to the result provided by Kontostathis et al. [58] according to which the average precision across all documents is 47.7%. We can see that our unsupervised approach, together with the proposed set of features, performs reasonably well respect to the values of the supervised C4.5 decision tree,. We can further observe that the supervised approach gets the precision and recall values of 0.60 and 0.40 respectively on 1.000 dataset size, while our model reports an average precision and recall of respectively 0.72 and 0.58 with an average F1 score of 0.64. For completeness, we report that a true positive rate of 78.5% has been reached in [70] when the positive posts were overrepresented in the training dataset.

6.3.3 Results on Youtube dataset

We tested our model also on the dataset referred by the paper from M. Dadvar et al. [71]. The dataset represents 3.462 comments from YouTube crawled in 2012-2013. The activities of users have been collected over 4 months (April-June 2012), together with profile information. A total of about 54.000 manually annotated comments over YouTube has been analyzed. Finally, the dataset has about 3.045 posts, where 419 posts contain cyber bullying traces. The dataset is unbalanced and so we adopted the same previous technique based on ran-

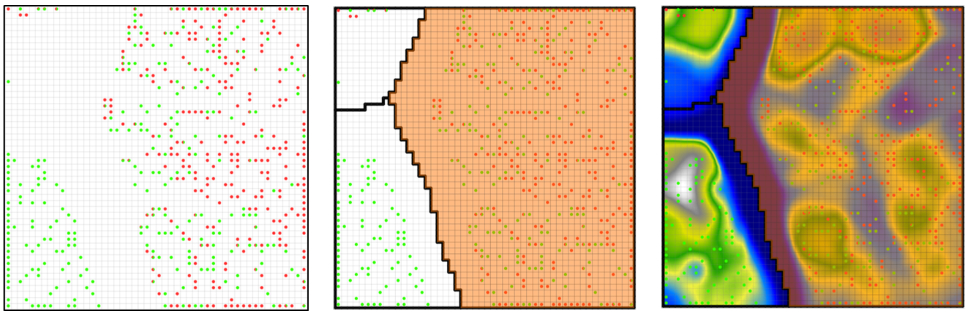


Figure 6.7: Class distribution (left) over the GHSOM winners neurons with related K-means clustering (center) and smoothed density map (right). Red dots mean cyber bully traces while green dots mean non bully traces (Youtube dataset).

Table 6.4: Results obtained on Youtube dataset.

Precision	**Accuracy**	**Recall**	**F1**	**Method**
0.60	0.69	0.94	0.74	GHSOM

dom under sampling. Also for this dataset we applied K-fold cross validation with K=10. For testing data we used 450 negative classes and 369 positive classes. Authors adopted a set of 11 features in three different categories and three machine learning methods, which use labelled training data: a Naive Bayes classifier, a classifier based on decision trees (C4.5) and Support Vector Machines (SVM) with a linear kernel, obtaining at the best 0,72 AUC score (ROC curve) with hybrid Naive Bayes classifier. Our results, based on unsupervised approach are shown visually in fig. 6.7. The precision, recall and F1 score are shown in table 6.4.

Some considerations must be done on the different results obtained for the YouTube dataset. First, user comments on YouTube are longer than the one usually posted in FormSpring. Therefore textual analysis and some related syntactical features perform differently on this dataset. According to our tests, in the YouTube case, it is possible to observe a more efficient clustering

Table 6.5: Results obtained on Twitter dataset.

Precision	**Accuracy**	**Recall**	**F1**	**Method**
0.81	0.72	0.26	0.4	GHSOM
-	0.67	-	-	Naive Bayes

of no bully posts, with respect to the FormSpring case. The selected cluster for the bully posts has a good recall value, but a lower precision result respect to the FormSpring test. The global measure F1 increased in this case from 0.64 of the FormSpring dataset to 0.74.

6.3.4 Results on Twitter dataset

Twitter allows users to write a maximum of 140 characters, so people are forced to share only essential information. This limitation (short documents) represents a big challenge in the text analysis since the most common adopted techniques perform better with long documents. Twitter users commonly adopt an informal language in posting messages, with many slang words and acronyms, also due to the limitation imposed by the platform.

We tested our unsupervised model with the Twitter dataset, adopted in [72], where authors, using a Naive Bayes classifier, reached an accuracy of 67.3% (see table 6.5). Here weak results of our model in recall and F1 score can be attributed to the dataset origin, that is much more related to general Sentiment Analysis tasks rather than cyber bullying.

For the Twitter case we conducted also a different qualitative test. We reused the previous trained GHSOM for the YouTube dataset, and tested it to see if our model can be generalized to classify a real Twitter stream of data. We collect 1.000 tweets during the summer of 2015, without any filtering, except for the language (En-

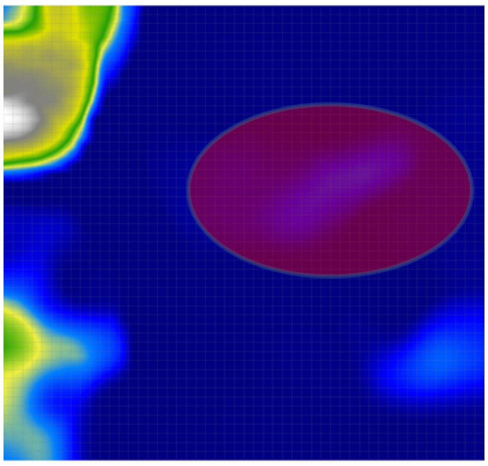


Figure 6.8: Density of the BMU for the a real twitter data set phase using a smoothed data histogram.

Table 6.6: Excerpt of tweets belonging to cluster identified in fig. 6.8.

Tweets sample in identified cluster (real test stream)
@*** wtf u literally post pics of u s**tting
@*** you're so fu**ing annoying lmao
@*** I'm out, I'm done wasting time with yo bit** ass

glish). In figure 6.8 we can see the smoothed data histogram of the network, showing the density (clearer areas) of BMU (Best Matching Unit). The real test dataset is unbalanced (statistically the percentage of cyber bullying posts in a twitter stream is between the 4% and 7% of the whole dataset). In order to visually see which are the identified clusters discovered by our model, we can observe the density map in fig. 6.8.

It's possible to verify how the tweets listed in table 6.6 (partially censored with the * char) can be considered possible candidates to bully traces, and as we expected there is an error rate that is proportional to the results obtained in the training stage.

6.4 Conclusions

In this chapter we have shown concrete results obtained applying our proposed model, labelled DBN-W2V, to two different datasets of movie reviews in Italian language, and we have also compared our model to some reference models, often adopted in this field. From our studies we can observe that deep learning can be considered an effective approach to general Sentiment Analysis tasks and polarity detection activity, even if it is really important to achieve an efficient text representation for input data, also able to keep semantic properties of textual data in the final vectorized representation.

For the cyber bullying detection task, we conducted some experiments on datasets taken from literature, like those coming from FormSpring and YouTube platforms, and also on a real data stream, collected from Twitter. Results indicate that our model achieves reasonable performance and could be usefully applied to build concrete monitoring applications to mitigate the heavy social problem of cyber bullying. Indeed, there is plenty of room for improvement on these techniques (as sarcasm identification) in order to achieve better results.

Chapter 7

Conclusions

7.1 Final remarks

In this research we have investigated the issue of defining and evaluating novel methods for Sentiment Analysis, also having as a target the Italian language, and focusing on machine learning techniques, based both on semi supervised and unsupervised approaches.

In chapter II we presented first a general study on the Sentiment Analysis task, its formal definitions and its complexities, exposing a taxonomy of the currently applied techniques in literature. Our work represents, at the best of our knowledge, the first attempt to solve this particular kind of problem for the Italian language, using a deep learning approach.

We proposed, in chapter III, to adopt a Deep Belief Network, together with a vectorized representation of textual input using the recent Word2Vec algorithm. We also analyzed and suggested a possible structure of the neural network in terms of its configuration layers, neurons type, error functions, classifier, etc. We built from scratch a vectorized representation of the data sets in Italian language, using the corpus (Paisá) of documents provided by CNR, and provides also 2 new different datasets of movie reviews, still in Italian language. From our studies we can observe that deep learning can be considered an effective approach to general Sentiment Analysis tasks and polarity detection activity, even if it's really important to achieve an efficient text representation for input data, also able to keep semantic properties of textual data in the final vectorized representation. For this purpose the Word2Vec algorithm is actually well suited, but it's also necessary to provide some fine tuned text pre-processing activities with Natural Language Processing techniques, that must be customized for the targeted language. Nevertheless, the proposed semi-supervised approach has obtained promising results respect to other methods from literature.

Furthermore, we also have considered, as an extreme Sentiment Analysis task (i.e. finding extreme negative polarities in sentences), the problem of automatic detection of cyber bullying traces, proposing a new model based on a set of hand-crafted features to be used with an unsupervised model of neural network, the Growing Hierarchical Self Organizing Map. Our first step was to understand the source and nature of the problem as a social phenomenon in order to identify the main aspects and which features could be identified to automatically detect harassing posts over social networks. More in details, we investigated the identification of possible attributes in posting activity and common behaviours of potential *cyber bullies* on line. Our consequent proposal, described in chapter IV, has been to define a novel set of hand crafted features that aim to model extreme negative contents (in terms of sentiment polarity) over social networks and subsequently to classify these contents using a Growing Hierarchical Self Organizing Map, in an unsupervised approach. We compared our results with different data sets taken from literature and related to different social networks (as Twitter, YouTube and FormSpring.me), outperforming some previous methods as SVM, (see chapter VI). Our remarks on this activity is that these kind of specific problems (i.e. cyber bullying) must be treated, with a needed a-priori knowledge on the model, in order to catch as much as possible aspects and peculiarities of this dangerous phenomenon, while is still possible and effective to adopt an unsupervised approach in order to automatically classify and cluster processed data, coming also from social networks. About this, in the Appendix A section we proposed also an architecture dedicated to the activities of analysis of big data, as the ones produced by social networks like Twitter, based on the Lambda structure.

7.2 Future works

The integration of social studies into a software framework able to monitor potential dangerous activities could lead the way towards the tackling of this increasing digital misbehaviour. We underline here the importance of a possible unsupervised approach, that could be more feasible in social networks scenarios, due to the huge data production.

A possible future track can be the integration of Sentiment Analysis tools and solutions, as the one proposed in this research, directly into mobile applications, that are normally used to post contents on social networks (i.e. Twitter mobile app), to preventively inhibit users from sending harmful text.

Usually, cyber bullying contents consist also of photos or videos containing harmful images (i.e. photos of a naked girl). In these cases it's possible to adopt computer vision techniques to discover such contents, even if this represents a real challenging task. Sentiment Analysis models and applications could be a real good tool to integrate with such algorithms, in order to enforce detection performances and try to mitigate this increasing dangerous social phenomenon.

Appendix A

An Architecture proposal for Sentiment Analysis

Parts of this section were originally published as Di Capua M., Di Nardo E, Petrosino A. "*An Architecture for Sentiment Analysis in Twitter*". International Conference on E-learning, ISSN: 2367-6698, Berlin, Germany. 2016.

A.1 Introduction

The definition and modelling of an architecture dedicated to the activities of analysis of big data, as the ones produced by social networks as Twitter, is currently still at an early stage of its development and consolidation. In this research we dealt with this kind of data for our polarity detection task, especially in the case of the Twitter platform (see paragraph 6.3.4). Unlike traditional data warehouse or business intelligence systems, whose architecture is designed for structured data, systems dedicated to big data work instead with semi-structured data, or so called "raw data", i.e. without a particular structure. It should also be pointed out that such systems should be able to allow processing and analysis of data not only in batch mode, but also in a real real-time fashion.

Nowadays a huge amount of data, daily produced by social networks, can be processed and analyzed for different purposes. These data are provided with several features, among which:

- dimension;
- peculiarities;
- source;
- reliability;

By the time the need to obtain the information and the way this information must be processed has changed. Until recently it was thought that the data should be first processed and subsequently made available, regardless of the time aspect. This type of processing is commonly called *batch processing*. Nowadays the amount of data is increased exponentially and now real-time processing is needed to get the most advantages from this data, in different fields, including Sentiment Analysis.

Actually batch models do not allow to work with the data in a real time fashion, due to the long time required by processing operations. Against the implementation of real-time processing architecture could lead to lower accuracy One possible solution is to merge the two concepts into a single architecture, capable of handling big data, but also with scalable and fast processing features.

A possible solution to this problem is the so called Lambda Architecture [3], a software architecture made by 3 different levels, as showed in figure A.1:

- Batch layer;
- Speed layer;
- Serving layer;

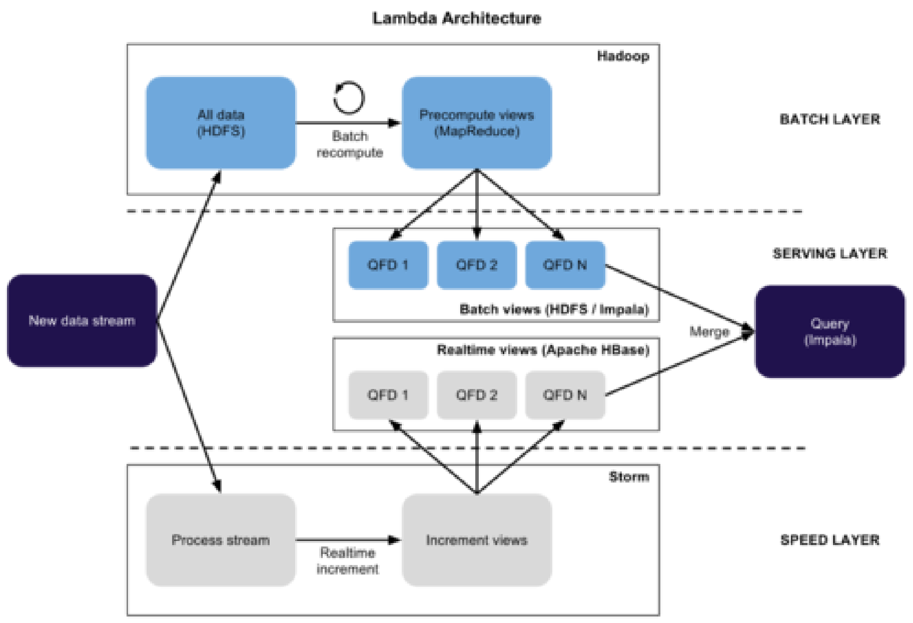


Figure A.1: Layer organization of a Lambda Architecture [3].

A.1.1 Batch Layer

This level is responsible to store the input data in a master dataset structure. These data are periodically processed, generally with a Map Reduce approach [73]. The batch layer precomputes results using a distributed processing system that can handle very large quantities of data. The batch layer aims at perfect accuracy by being able to process all available data when generating views. This means it can fix any errors by recomputing based on the complete data set, then updating existing views. Output is typically stored in a read-only database, with updates completely replacing existing precomputed views. Several frameworks can be used at this level, but the two most adopted are:

a) **Apache Hadoop** can be used for the Map Reduce operations. It uses a proprietary file system called Hadoop Distributed File System (HDFS), which reads and writes using customizable I/O drivers. It's a distributed file system with also failure-tolerant capabilities. It's available in different language implementations as Java and Python.

b) **Apache Spark** can be used for the Map Reduce operations but also for general-purpose task. It's able to perform on memory operation and it supports also HDFS file system HDFS and the Hadoop drivers. It's distributed and fault tolerant. It came with different libraries for the view data generation, micro-batching, machine learning and also graph based analysis operations.

A.1.2 Speed Layer

This layer processes data streams in real time. This layer sacrifices throughput as it aims to minimize latency by providing real-time views into the most recent data. Essentially, the speed layer is responsible for filling the "gap" caused by the batch layer's lag in providing views based on the most recent data. This layer's views may not be as accurate or complete as the ones eventually produced by the batch layer, but they are available almost immediately after data is received, and can be replaced when the batch layer's views for the same data become available. As for the batch layer several frameworks, based on stream processing technologies, can be used at this level. The most adopted are:

a) **Apache Storm** is based on the idea of streaming computation, i.e. processing new data individually. It provides support for micro-batching operations, collecting new data in small data set and then executing operations on the whole dataset with the Trident API. It's distributed and it uses Apache Zookeeper for this task. The core abstraction in Storm is the "stream". A stream is an unbounded sequence of tuples. Storm provides the primitives for transforming a stream into a new stream in a distributed and reliable way. For example, you may transform a stream of tweets into a stream of trending topics. The basic primitives Storm provides for doing stream transformations are "spouts" and "bolts". Spouts and bolts have interfaces that you implement to run your application-specific logic. A spout is a source of streams. For example, a spout may connect to the Twitter API and emit a stream of tweets. A bolt consumes any number of input streams, does some processing, and possibly emits new streams. Complex stream transformations, like computing a stream of trending topics from a stream of tweets, require multiple steps and thus multiple bolts. Bolts can do anything from run functions, filter tuples, do streaming aggregations, do streaming joins, talk to databases, and more. Networks of spouts and bolts are packaged into a "topology" which is the top-level abstraction that you submit to Storm clusters for execution. A topology is a graph of stream transformations where each node is a spout or bolt. Edges in the graph indicate which bolts are subscribing to which streams. When a spout or bolt emits a tuple to a stream, it sends the tuple to every bolt that subscribed to that stream.

b) **Apache Spark Streaming** is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Data can be ingested from many sources like Twitter, and can be processed using complex algorithms expressed with high-level functions like map, reduce, join and window. Finally, processed data can be pushed out to filesystems, databases, and live dashboards. It's also possible to apply Spark's machine learning and graph processing algorithms on data streams.

A.1.3 Serving Layer

This layer takes care of the output task, joining the results of the Batch and Speed layer, in order to obtain a single view of the data. A critical task at this level is related to data synchronization activities. It's worth here to integrate a storage engine capable of executing random reads, bulk writes, with a low latency and also in a distributed manner. Typically these kind of operations are well performed in storage organized in key/value, such as: ElephantDB, Redis, Hbase, Druid, and Elastic search.

A.2 Sentiment Analysis Application

We have seen previously formal definitions of Sentiment Analysis, also called opinion mining, as the field of study that analyses people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes [23].

We also discussed the two main applied techniques for sentiment analysis activity: machine learning based and lexicon based. Few research studies have also combined this two methods to gain relatively better performance and results. However, both the approaches used in literature so far have shown two fundamental problems:

1) In the context of machine learning techniques along with the support of techniques derived from NLP (Natural Language Processing), the algorithms developed work usually on data that are rendered in formats tailored to the particular algorithm developed. For example in the area of text classification, one commonly adopted solution is to use unordered lists of words (*bag of words*), ignoring thereby the relations between the words of a sentence and the grammatical structure of the sentence itself. Especially in the field of sentiment analysis, the correlation between the words close together can dramatically change the meaning of a sentence in context. In summary a major problem of the solutions adopted NLP concerns about simplifying assumptions of phrases and language analyzed.

2) Another fundamental problem relates to the representation of the features. The secret of many analytical systems depends on the type of representation of the features used, such as the choice of named entities (i.e. persons or organizations), or the use of part of speech tagging. The use of ad hoc features involves consuming computational resources and makes the system or algorithm developed less flexible for purposes other than those for which it was realized.

Therefore, in some scenarios, it's desirable to adopt a general representation of sentences, but without losing information on the grammatical structure in which these sentences are presented [74].

In this work we have explored an architecture capable of processing large amounts of data from Twitter. For the processing phase, in order to perform sentiment analysis on data, we propose to integrate in the same software architecture, some software components that implements techniques of machine learning together with NLP algorithms.

More in details we propose to adopt a lambda architecture and to use, at the batch level of the architecture, a recursive neural network model, based on a pre-trained Kohonen SOM (Self Organized Map).

A SOM is a neural network that is able to map data input received into different corresponding regions, with different regions (clusters) having different response characteristics for corresponding input mode. Generally, SOM has proven to be an efficient and suitable documents clustering method. It can map documents onto two-dimensional diagram to show the relationship between the different documents. SOM can depict text in more figurative and better visual way. Text Clustering is a high-dimensional application and closely related to the semantic features. The above characteristics of SOM make it very suitable for text clustering and subsequent sentiment analysis. By inputting a document, usually using a vector representation of its words, the neurons representing the pattern class-specific (sentiment related) in the output layer will have the greatest response.

Recursion is obtained by applying the same neural network at each node of the grammatical structure, represented by the semantic tree of the sentence. Grammatical structures, where used, can solve the problem of so-called *propositional attachment*. The details on the machine learning algorithms and techniques applied here is not covered in this appendix, but we simply focus our attention on the architectural components of the solution (see figure A.2).

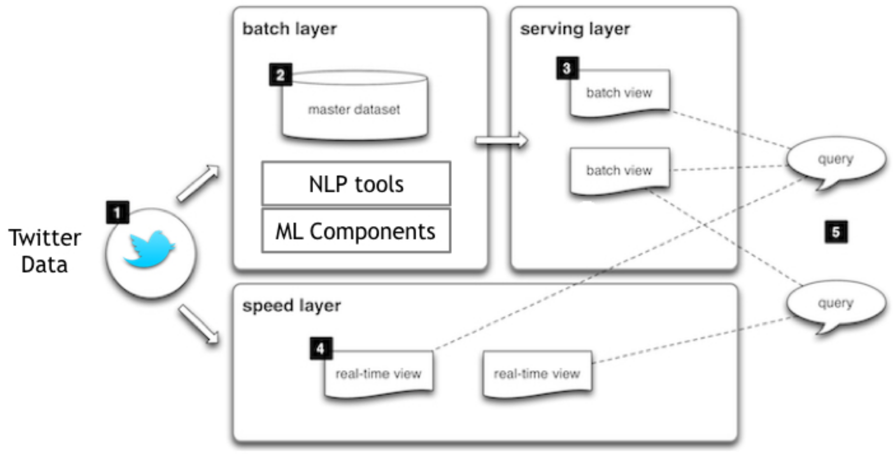


Figure A.2: Modified Lambda Architecture for Sentiment Analysis tasks.

The models of recursive networks, not only are able to predict this type of structure of sentences, but can also learn the meaning of a composition of words within a tweet. Thus some problem can be solved problems such as learning vector of features for variable input size, without ignoring the structure or sequence of words, as it appears in the sentence. Most of these deep models are able to learn also compositional semantics, simply starting from training data without having any manual description of features.

Preliminary results testing this architecture on tweets streaming data seems to be promising. The tweets are obtained and submitted to the nodes of the architecture using the Twitter Streaming API, with some applied filtering criteria only on the region and on the language (ENG) of the authors. The architecture is able to process at a 600k tweets/sec per each node, using Apache

Spark as integrated component at the batch layer.

A.3 Conclusions

Applying sentiment analysis techniques to mine the huge amount of unstructured data in social networks has become an important research problem. Business organizations and research institutes are putting their efforts to improve techniques for sentiment analysis. Although, some algorithms have been used in sentiment analysis with good results, there are still no techniques able to resolve all the current challenges.

Focusing in this appendix on the architectural aspects of Sentiment Analysis tasks, we discussed here a software system that receives events as big data (tweets), archives them, performs both offline and real-time computations for sentiment assessment, and merges the results of those computations into coherent information. All of this needs to happen at the scale of millions events per second.

Further work is needed on further improving both the accuracy of the sentiment classification (as discussed in this thesis) and also improving the rate at which this huge amount of information must be processed.

Appendix B

Publications

Here it is reported a list of publications produced during the research period underneath this work of thesis:

- Di Capua M., Di Nardo E, Petrosino A. *"Unsupervised Cyber Bullying Detection in Social Networks"*. In Proceedings of ICPR, 23rd International Conference on Pattern Recognition. Cancun, Mexico. December 2016.
- Di Capua M., Di Nardo E, Petrosino A. "*An Architecture for Sentiment Analysis in Twitter*". International Conference on E-learning, ISSN: 2367-6698, Berlin, Germany. September, 2015.
- Di Capua M., Petrosino A. "*Semi supervised Deep Learning Approach to Deal with Data Uncertainty in Sentiment Analysis*". In Proceedings of WILF 2016, 11th International Workshop on Fuzzy Logic and Applications. LNAI 10147. Naples, Italy. 2016.
- Di Capua M., A. Petrosino. *"Unsupervised Sentiment Analysis using Deep Neural Architectures"*. Information Sciences Journal. Informatics and Computer Science Intelligent Systems Applications. Elsevier. (submitted).

Appendix C

Software and Configurations

Here it is reported a list of the main software tools used in this thesis:

- The framework adopted for the implementantion of the deep belief network model is **deeplearning4j**[1], and open source softwarethat can used also for distributed learning activities and that can be integrated also with Apache Hadoop ed Apache Spark platforms. Deeplearning4j is built in Java and Scala (functional language programming), and has been designed for production environment.
- The implementation of the **Word2Vec** algorithm is available as a library built-in inside the framework deeplearing4j.
- The POS tagging (ITA) component has been realized using the auxiliary open source framework **Apache Open NLP**[2]. This framework supports common functions of the NLP area, as tokenization, sentence segmentation, grammar analysis, entity extraction, and co-reference.
- In this work we adopted 2 different libraries for the

[1]www.deeplearning4j.org
[2]https://opennlp.apache.org/

scientific processing, **ND4J** ed **ND4S**, both designed in production environment with low level of RAM consuming and optimized capacity utilization of the machine's computational resources (GPU).

- For the algebraic processing component it was used **OpenBLAS** library. BLAS stands for Basic Linear Algebra Subprograms. BLAS provides standard interfaces for linear algebra, including BLAS1 (vector to vector operations), BLAS2 (matrix to vector operations), and BLAS3 (matrix to matrix operation).

Bibliography

[1] S. Zhou, Q. Chen, X. Wang, and X. Li, "Hybrid deep belief networks for semi-supervised sentiment classification," 2014.

[2] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data," *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1331–1341, 2002.

[3] N. Marz and J. Warren, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Greenwich, CT, USA: Manning Publications Co., 1st ed., 2015.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[5] R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning, "Parsing natural scenes and natural language with recursive neural networks," 2011.

[6] J. Wiebe, T. Wilson, and M. Bell, "Identifying collocations for recognizing opinions," in *In Proc. ACL-01 Workshop on Collocation: Computational Extraction, Analysis, and Exploitation*, pp. 24–31, 2001.

[7] T. Nasukawa and J. Yi, “Sentiment analysis: Capturing favorability using natural language processing,” in *Proceedings of the 2Nd International Conference on Knowledge Capture*, K-CAP ’03, (New York, NY, USA), pp. 70–77, ACM, 2003.

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

[9] K. Dave, S. Lawrence, and D. 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, (New York, NY, USA), pp. 519–528, ACM, 2003.

[10] V. Hatzivassiloglou and K. R. McKeown, “Predicting the semantic orientation of adjectives,” in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL ’98, (Stroudsburg, PA, USA), pp. 174–181, Association for Computational Linguistics, 1997.

[11] M. A. Hearst, “Automatic acquisition of hyponyms from large text corpora,” in *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING ’92, (Stroudsburg, PA, USA), pp. 539–545, Association for Computational Linguistics, 1992.

[12] J. M. Wiebe, “Identifying subjective characters in narrative,” in *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING ’90, (Stroudsburg, PA, USA), pp. 401–406, Association for Computational Linguistics, 1990.

[13] J. M. Wiebe, “Tracking point of view in narrative,” *Comput. Linguist.*, vol. 20, pp. 233–287, June 1994.

[14] R. F. Bruce and J. M. Wiebe, “Recognizing subjectivity: A case study in manual tagging,” *Nat. Lang. Eng.*, vol. 5, pp. 187–205, June 1999.

[15] P. 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, (Stroudsburg, PA, USA), pp. 417–424, Association for Computational Linguistics, 2002.

[16] S. Das and M. Chen, “Yahoo! for amazon: Extracting market sentiment from stock message boards,” in *In Asia Pacific Finance Association Annual Conf. (APFA)*, 2001.

[17] S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima, “Mining product reputations on the web,” in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’02, (New York, NY, USA), pp. 341–349, ACM, 2002.

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

[19] R. Tong, “An operational system for detecting and tracking opinions in on-line discussions,” in *Working Notes of the SIGIR Workshop on Operational Text Classification*, (New Orleans, Louisianna), pp. 1–6, 2001.

[20] J. Wiebe, T. Wilson, R. Bruce, M. Bell, and M. Martin, "Learning subjective language," *Comput. Linguist.*, vol. 30, pp. 277–308, Sept. 2004.

[21] E. Riloff and J. Wiebe, "Learning extraction patterns for subjective expressions," in *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, (Stroudsburg, PA, USA), pp. 105–112, Association for Computational Linguistics, 2003.

[22] H. Yu and V. Hatzivassiloglou, "Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences," in *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, (Stroudsburg, PA, USA), pp. 129–136, Association for Computational Linguistics, 2003.

[23] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, (New York, NY, USA), pp. 168–177, ACM, 2004.

[24] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, (Stroudsburg, PA, USA), pp. 151–161, Association for Computational Linguistics, 2011.

[25] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural*

Language Learning, EMNLP-CoNLL ’12, (Stroudsburg, PA, USA), pp. 1201–1211, Association for Computational Linguistics, 2012.

[26] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. P. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *EMNLP*, 2013.

[27] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[28] R. Collobert, “Deep learning for efficient discriminative parsing,” in *International Conference on Artificial Intelligence and Statistics*, 2011.

[29] B. Pang and L. Lee, “A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts,” in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, p. 271, Association for Computational Linguistics, 2004.

[30] F. Benamara, C. Cesarano, A. Picariello, D. R. Recupero, and V. S. Subrahmanian, “Sentiment analysis: Adjectives and adverbs are better than adjectives alone,” in *ICWSM*, Citeseer, 2007.

[31] A. Balahur and M. Turchi, “Multilingual sentiment analysis using machine translation?,” in *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA ’12, (Stroudsburg, PA, USA), pp. 52–60, Association for Computational Linguistics, 2012.

[32] K. Dashtipour, S. Poria, A. Hussain, E. Cambria, A. Y. A. Hawalah, A. Gelbukh, and Q. Zhou, "Multilingual sentiment analysis: State of the art and independent comparison of techniques," *Cognitive Computation*, vol. 8, no. 4, pp. 757–771, 2016.

[33] E. Boiy and M.-F. Moens, "A machine learning approach to sentiment analysis in multilingual web texts," *Information Retrieval*, vol. 12, no. 5, pp. 526–558, 2009.

[34] K. Denecke, "Using sentiwordnet for multilingual sentiment analysis," in *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, pp. 507–512, April 2008.

[35] M. Bautin, L. Vijayarenu, and S. Skiena, "International sentiment analysis for news and blogs.," in *Association for the Advancement of Artificial Intelligence, Book chapter*, 2008.

[36] Y. Seki, D. K. Evans, L.-W. Ku, L. Sun, H.-H. Chen, and N. Kando, "Overview of multilingual opinion analysis task at ntcir-7.," in *Proceedings of the Seventh NTCIR Workshop*, 2008.

[37] U. Mirchev and M. Last, "Multi-document summarization by extended graph text representation and importance refinement," *Innov Doc Summ Tech Revolut Knowl Underst Revolut Knowl Underst*, vol. 28, 2014.

[38] D. K. Evans, L.-W. Ku, Y. Seki, H.-H. Chen, and N. Kando, "Opinion analysis across languages: An overview of and observations from the ntcir6 opinion analysis pilot task," in *Proceedings of the 7th International Workshop on Fuzzy Logic and Applications: Applications of Fuzzy Sets Theory*, WILF '07,

(Berlin, Heidelberg), pp. 456–463, Springer-Verlag, 2007.

[39] O. Chapelle, B. Schlkopf, and A. Zien, *Semi-Supervised Learning.* The MIT Press, 1st ed., 2010.

[40] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *J. Mach. Learn. Res.*, vol. 10, pp. 1–40, June 2009.

[41] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[42] J. Hastad, "Almost optimal lower bounds for small depth circuits," in *RANDOMNESS AND COMPUTATION*, pp. 6–20, JAI Press, 1989.

[43] J. Hastad and M. Goldmann, "On the power of small-depth threshold circuits," *COMPUTATIONAL COMPLEXITY*, vol. 1, pp. 610–618, 1991.

[44] Y. Bengio, P. Lamblin, D. Popovici, and Q. Hugo Larochelle. Universite de Montreal, "Greedy layer-wise training of deep networks," in *NIPS*, MIT Press, 2007.

[45] Y. Bengio and O. Delalleau, "On the expressive power of deep architectures," in *In ALT 2011*, 2011.

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

[47] M. Ranzato, B. Y-lan, and Y. LeCun, "Sparse feature learning for deep belief networks," in *Advances in neural information processing systems*, pp. 1185–1192, 2008.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neu-

ral networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[49] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS 2010). Society for Artificial Intelligence and Statistics*, 2010.

[50] J. W. Patchin and S. Hinduja, "Bullies move beyond the schoolyard a preliminary look at cyberbullying," *Youth violence and juvenile justice*, vol. 4, no. 2, pp. 148–169, 2006.

[51] S. Nalini, "A survey on datamining in cyber bullying," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, no. 7, 2014.

[52] A. Sourander, K. Brunstein, M. Ikonen, and et al, "Psychosocial risk factors associated with cyberbullying among adolescents: A population-based study," *Archives of General Psychiatry*, vol. 67, no. 7, pp. 720–728, 2010.

[53] R. M. Kowalski, S. P. Limber, and P. W. Agatston, *Cyberbullying: Bullying in the Digital Age.* Wiley Publishing, 2nd ed., 2012.

[54] N. E. Willard, *Cyberbullying and Cyberthreats: Responding to the Challenge of Online Social Aggression, Threats, and Distress.* Research Publishers LLC, 2nd ed., 2007.

[55] D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards, "Detection of harassment

on web 2.0," *Proceedings of the Content Analysis in the WEB*, vol. 2, pp. 1–7, 2009.

[56] M. Dadvar and F. De Jong, "Cyberbullying detection: a step toward a safer internet yard," in *Proceedings of the 21st International Conference on World Wide Web*, pp. 121–126, ACM, 2012.

[57] K. Dinakar, R. Reichart, and H. Lieberman, "Modeling the detection of textual cyberbullying.," *The Social Mobile Web*, vol. 11, p. 02, 2011.

[58] A. Kontostathis, K. Reynolds, A. Garron, and L. Edwards, "Detecting cyberbullying: query terms and techniques," in *Proceedings of the 5th annual acm web science conference*, pp. 195–204, ACM, 2013.

[59] J.-M. Xu, K.-S. Jun, X. Zhu, and A. Bellmore, "Learning from bullying traces in social media," in *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pp. 656–666, Association for Computational Linguistics, 2012.

[60] D. Terrana, A. Augello, and G. Pilato, "Automatic unsupervised polarity detection on a twitter data stream," in *Semantic Computing (ICSC), 2014 IEEE International Conference on*, pp. 128–134, IEEE, 2014.

[61] T. Kohonen, ed., *Self-organizing Maps.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997.

[62] E. Pampalk, G. Widmer, and A. Chan, "A new approach to hierarchical clustering and structuring of data with self-organizing maps," *Intell. Data Anal.*, vol. 8, pp. 131–149, Apr. 2004.

[63] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st ed., 2000.

[64] M. Baroni, S. Bernardini, F. Comastri, L. Piccioni, A. Volpi, G. Aston, and M. Mazzoleni, "Introducing the la repubblica corpus: A large, annotated, tei (xml)-compliant corpus of newspaper italian," *issues*, vol. 2, pp. 5–163, 2004.

[65] M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta, "The wacky wide web: A collection of very large linguistically processed webcrawled corpora. language resources and evaluation," 2009.

[66] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 111–147, 1974.

[67] K. Kamvar, S. Sepandar, K. Klein, D. Dan, M. Manning, and C. Christopher, "Spectral learning," in *International Joint Conference of Artificial Intelligence*, Stanford InfoLab, 2003.

[68] R. Collobert, F. Sinz, J. Weston, and L. Bottou, "Large scale transductive svms," *Journal of Machine Learning Research*, vol. 7, no. Aug, pp. 1687–1712, 2006.

[69] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Inf. Sci.*, vol. 250, pp. 113–141, 2013.

[70] K. Reynolds, A. Kontostathis, and L. Edwards, "Using machine learning to detect cyberbullying," in *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, vol. 2, pp. 241–244, IEEE, 2011.

[71] M. Dadvar, D. Trieschnigg, and F. de Jong, "Experts and machines against bullies: A hybrid approach to detect cyberbullies," in *Canadian Conference on Artificial Intelligence*, pp. 275–281, Springer, 2014.

[72] H. Sanchez and S. Kumar, "Twitter bullying detection," in *NSDI*, pp. 15–22, Berkeley, CA, USA: USENIX Association, 2012.

[73] J. Berlińska and M. Drozdowski, "Scheduling divisible mapreduce computations," *J. Parallel Distrib. Comput.*, vol. 71, pp. 450–459, Mar. 2011.

[74] O. İrsoy and C. Cardie, "Deep recursive neural networks for compositionality in language," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), pp. 2096–2104, Curran Associates, Inc., 2014.