



**POLITECNICO DI MILANO  
POLO TERRITORIALE  
DI COMO**

Corso di Laurea Magistrale in Ingegneria Informatica  
Dipartimento di Elettronica e Informazione

**Power of time window in predicting hashtags  
while typing a tweet**

Supervisor: Prof. Emanuele Della Valle  
Co-Supervisor: Prof. Davide Eynard

Master Thesis by: Farimah Fanaei  
Student ID N.816079

A.Y. 2015/2016

## **Acknowledgement**

First of all, I would like to thank Politecnico Di Milano for giving me this great opportunity to study this master and live in this beautiful country.

Deeply, I would like to express my special thanks of gratitude to my supervisor Prof. Emanuele Della Valle, for enthusiastic support, useful comments, remarks and engagement through the learning process of this master thesis.

I am also very grateful to my co-supervisor Prof. Davide Eynard, who supported me and provided very useful suggestions about this thesis. I am very much indebted to him for the success of my thesis work.

Last but not least, I would like to thank my lovely family and my friends Maryam, Atusa, Masoumeh, Sepehr, Reyhaneh, Armin, Elmira, Mahdi and Arash for their kind support through my life.

## Abstract:

Micro-blogging as a short text reports like tweets is widely used recently, processing and extracting the knowledge from this huge volume of information stream has attracted many attentions. In general Text Analysis and Text Processing is an essential task in Natural Language Processing and also a major application field in Machine Learning area, there has been wide range of approaches proposed both in NLP and ML. Central part of solutions in this regard is around finding data patterns and study the features of the data. Finding the accurate and proper form of features can be significantly important in data processing, for text processing previous works, usually exploit only human-designed features, such as dictionaries, knowledge bases and special tree kernels, although currently advanced approaches get involved with deeper semantical layers rather than only retrieving features from text, these approaches appear to have a great progress in boosting the Text Classification performance.

The importance of social networks like twitter and their universal propagation in the last few years represents one of the most pervasive phenomena of the recent computer and data science society, keeping track of predictive behavior of hashtags in tweets have been an interesting topic that can be reached by many approaches. Although applying Machine Learning and Information Retrieval on short documents like tweets is not an easy task, due to few numbers of features in each sample which can reduce accuracy of the final predictive analysis. The solution can be using these numerable features efficiently by extracting latent properties and processing semantical aspects of the features.

Although it is needless to mention that text processing on stream data can be even more challenging, on one hand we usually do not have a fixed number of features, on the other hand these features can have inconstant importance at different time windows so we need to verify this fact at every stage by updating ML methods and fitting the model with recent training set, this can boost degree of accuracy in pattern recognition and enhance the precision in predicting the output values.

In this thesis we work on a dataset with more than 2 million of tweets collected in more than 4 months from Expo 2015 in Milan. The general goal is predicting second hash-tags considering its relativeness with upcoming events and features at different time windows. With the purpose of comparing different ML classification methods there was 2 main hypotheses: 1. Effect of applying different time window lengths on the accuracy of the classification on stream data. 2. Applying dimension reduction methods only on the prediction values can boost the final score of classification result. As usual the data processing task will start with Preprocessing of the data as the first step, eliminate all outliers and irrelevant elements of data, and evoking useful and informative form of words then converting them into standard feature vectors can be briefly consider as what we mean by preprocessing. By Finishing the preprocessing steps and achieving the proper informative vector form of the data acceptable for machine learning algorithms, next is to step into ML world, here we compare 5 different Classification results with different time windows where we applied a 2-layer feed-forward **Neural Network** algorithm to reduce the dimensionality of the outputs. Finally, the result approved both hypothesis and face an interesting conclusion about the effect of preprocessing to enhance the final result.

|  |     |
|--|-----|
| <b>Acknowledgement</b> .....                                     | I   |
| <b>Abstract:</b> .....   | II  |
| List of Figures .....  | V   |
| List of Table .....  | VI  |
| List of equations .....  | VII |
| <b>Chapter 1</b> .....   | 8   |
| <b>1. Introduction</b> .....                                     | 8   |
| <b>1.1- Problem Statement</b> .....                              | 9   |
| <b>1.2 Original Contribution</b> .....                           | 10  |
| <b>1.3 Structure of the thesis</b> .....                         | 12  |
| <b>Chapter 2</b> .....   | 13  |
| <b>2. State of Art</b> .....                                     | 13  |
| <b>2.1Data Preprocessing</b> .....                               | 13  |
| <b>2.1.1 Text documents Preprocessing</b> .....                  | 15  |
| <b>2.1.1 Co-occurrence</b> .....                                 | 16  |
| <b>2.2 Feature Engineering</b> .....                             | 18  |
| <b>2.2.2 Features Selection</b> .....                            | 20  |
| <b>2.3 Machine Learning:</b> .....                               | 22  |
| <b>2.3.1 Classification</b> .....                                | 26  |
| <b>2.3.2 Text Classification</b> .....                           | 28  |
| <b>2.3.2.1 Short-text Classification</b> .....                   | 33  |
| <b>2.4 NLP</b> .....   | 34  |
| <b>2.4.1 Word Embedding</b> .....                                | 35  |
| <b>2.5 Neural Networks</b> .....                                 | 37  |
| <b>2.5.1 multilayer Neural Networks</b> .....                    | 38  |
| <b>2.6.1 Machine learning on stream data</b> .....               | 40  |
| <b>Chapter 3</b> .....   | 42  |
| <b>3. Problem statement</b> .....                                | 42  |
| <b>3.1 Problem</b> .....   | 42  |
| <b>3.2 Case study</b> .....                                      | 44  |
| <b>Chapter 4</b> .....   | 47  |
| <b>4. Problem Solving</b> .....                                  | 47  |
| <b>4.1 System Architecture</b> .....                             | 47  |
| <b>4.2 Dataset</b> .....   | 50  |
| <b>4.3 Cleaning Data &amp; Text Preprocessing</b> .....          | 55  |
| <b>4.3.1 Stop Words</b> .....                                    | 55  |
| <b>4.3.2 Bag-of-words</b> .....                                  | 56  |
| <b>4.3.3 Transforming Text Documents into Vector Space</b> ..... | 57  |
| <b>4.4 Word2Vec</b> .....  | 59  |

|  |    |
|--|----|
| 4.4.1 Word2vec Architecture .....                          | 59 |
| 4.4.1.1 Continuous Bag-of-Words Model.....                 | 61 |
| 4.4.1.2 Continuous Skip-gram Model.....                    | 62 |
| 4.4.2 GloVe (Global Vectors for Word Representation) ..... | 63 |
| chapter5.....  | 64 |
| 5. Implementation Experience .....                         | 64 |
| 5.1 Multi-language tweets preprocessing .....              | 64 |
| 5.2 Data to Vector.....                                    | 67 |
| 5.3 fastFM .....   | 68 |
| 5.4 Cross-Validation.....                                  | 70 |
| 5.4.1 Cross-validation on stream data.....                 | 71 |
| chapter 6.....   | 73 |
| 6. Testing and Evaluation .....                            | 73 |
| 6.1 Baseline.....  | 74 |
| 6.1.1 Co-occurrence.....                                   | 74 |
| 6.1.2 Results .....  | 77 |
| 6.1.3 Conclusion.....                                      | 80 |
| 6.2 Experimental Tests.....                                | 80 |
| Chapter 7.....   | 85 |
| 7. Conclusion.....   | 85 |
| 7.1 Problem and adopted solutions.....                     | 85 |
| 7.2 limitations.....                                       | 87 |
| 7.3 Future Works .....                                     | 87 |
| References .....   | 88 |
| Appendix .....   | 91 |

## List of Figures

|   |    |
|---|----|
| Figure 1 d-dimensional feature space .....                                      | 8  |
| Figure 2 rectangular matrix .....   | 17 |
| Figure 3 square matrix .....  | 17 |
| Figure 4 representation of an utterance .....                                   | 18 |
| Figure 5 overview of fitting a model in ML .....                                | 23 |
| Figure 6 probability distribution estimation.....                               | 25 |
| Figure 7 Testing & Training in ML .....   | 30 |
| Figure 8 KNN classification method.....   | 31 |
| Figure 9 Naive Bayesian Method .....  | 32 |
| Figure 10 Support Vector Machine.....   | 33 |
| Figure 11 Modular Network to determine validity of a 5-gram example .....       | 36 |
| Figure 12 3-layer neural network.....   | 39 |
| Figure 13 Frequency of number of terms in tweets .....                          | 44 |
| Figure 14 System architecture overview.....                                     | 49 |
| Figure 15 Expo2015 Data Tables.....   | 50 |
| Figure 16 Frequency of tweets .....   | 51 |
| Figure 17 distribution of tweets for each user.....                             | 53 |
| Figure 18 hash-tag distribution in tweets .....                                 | 54 |
| Figure 19 word2vec architecture overview.....                                   | 62 |
| Figure 20 baseline(co-occurrence) prediction errors .....                       | 77 |
| Figure 21 Top-K prediction error rate.....                                      | 78 |
| Figure 22 comparison among top-k and co-occurrence.....                         | 79 |
| Figure 23 Performance comparison among different methods in machine learning .. | 83 |
| Figure 24 time window effect on the second hashtag prediction.....              | 84 |

## List of Table

|   |    |
|---|----|
| Table 1 Expo2015 Dataset Overview .....                         | 44 |
| Table 2 Most Frequent hashtags in Expo2015 .....                | 52 |
| Table 3 effect of dimension-reduction on error rate .....       | 81 |
| Table 4 effect of specialized preprocessing on error rate ..... | 81 |

## List of equations

|  |    |
|--|----|
| Equation 1 feature normalizing .....             | 20 |
| Equation 2 Linear function.....                  | 27 |
| Equation 3 logistic regression .....             | 27 |
| Equation 4 log function.....                     | 27 |
| Equation 5 Maximum Likelihood.....               | 27 |
| Equation 6 classification method .....           | 28 |
| Equation 7neural network hypothesis .....        | 39 |
| Equation 8 Compact NN activation formula.....    | 40 |
| Equation 9 recursive activation NN formula ..... | 40 |
| Equation 10 TFIDF/IDF formula .....              | 57 |
| Equation 11 normalized TFIDF .....               | 58 |

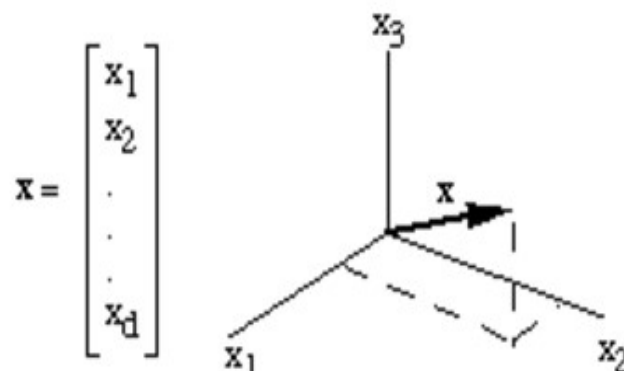


# Chapter 1

## 1. Introduction

Due to the high volume of incoming data in many available social networking services bring attention to extract valuable knowledge and latent data features. So many Machine Learning and Data Mining techniques have been proposed to draw data patterns and subsequently retrieve useful information in text documents. However, when we are facing a problem with stream data, efficiency of these methods strongly depends proper application and updating of these patterns and features. This problem is still an open research issue in the world of text classification (1).

Machine Learning and Information Retrieval has many important applications in today world, these technologies can be applied in different types of data as well as text data. However, they are not supposed to use pure textual data format directly for their statistical analysis and processing. Actually mathematical and statistical structure base techniques require numerical representation of features to be received as input which called Feature Vectors, an n-dimensional numerical vector form of input data samples. We can think of  $\mathbf{x}$  as being a point in a d-dimensional **feature space**<sup>1</sup>. By this process of feature measurement, we can represent an object or event abstractly as a point in feature space. (2)



*Figure 1 d-dimensional feature space*

1 [https://www.cs.princeton.edu/courses/archive/fall08/cos436/Duda/PR\\_model/f\\_vecs.htm](https://www.cs.princeton.edu/courses/archive/fall08/cos436/Duda/PR_model/f_vecs.htm)

So we are mapping our data points into these vector features although usually in text format of the data we face high dimensional vectors including some irrelevant features and the fact is that the data can also lie near a lower-dimensional. (Ghods 2006) So finding a small subset of most predictive features in a high dimensional feature space is an interesting issue (3). Information Retrieval proposes that word stems work well as representation units and doesn't concentrate on the words ordering for many tasks. This leads to an attribute value representation of text. Every single word  $W_i$  corresponds to a feature, and its value is equal to the number of times word  $W_i$  occurs in the document. [5]

In text classification recently many approaches use information gain, information about words correlation or L1 regularization in order to perform the feature selection. Although currently different deep learning methods using multi-layer neural network algorithms (Lai and Xu 2015) which can drive the syntactical relation among features rather than only retrieving them from text and are acting more efficiently in Text Classification tasks.

## 1.1- Problem Statement

In traditional text classification problems, the focus was generally on three topics: feature extraction, feature selection and using different types of machine learning algorithms (1). In feature extraction or feature engineering, the most widely used extraction method feature is the bag-of-words. Feature selection aims at reducing the feature space by throwing out some noisy features and improving the classification performance. The most common feature selection method is deleting the stop words (e.g., "of"). Advanced approaches use information gain, mutual information (Cover and Thomas 2012), or L1 regularization (Ng 2004) in order to select useful features. Finally, these features could be fed to Machine learning algorithms often use classifiers such as logistic regression (LR), naive Bayes (NB), and support vector machine (SVM). However, these methods have the data sparsity problem.

As an example Bag-of-words is a very well-known representation approach in object categorization area, widely used in text categorization also developed for image objects. Nevertheless, this method does not consider word orders in corpus and so cannot learn semantics of the words. However, these feature vectors are proper format of input data for machine learning algorithms for further prediction proposes but not the best case.

Recently rather than developing previous approaches, deep neural networks (Hinton and Salakhutdinov 2006) and representation learning (Bengio, Courville, and Vincent 2013) have been widely studied in many recent researches and they have advanced a new idea for solving the data sparsity problem. Also there have been many neural models introduced for learning word representations (4) (Bengio et al. 2003; Mnih

and Hinton 2007; Mikolov 2012; Collobert et al. 2011; Huang et al. 2012; Mikolov et al. 2013) and lately Convolutional Neural Networks (Lai and Xu 2015) with even higher performance in text classification.

Consequently, by follow recent research's trends and wide propagation of Neural Networks we realize the fact that by applying deeper multilayer Neural Networks one can capture syntactic relativeness and learn deeply representation of the words. Word Embedding is new neural representation of a word in text documents, so that there is an equivalent numerical vector for representing each word. Word embedding's one of the few currently successful applications of unsupervised learning. Naturally, every feed-forward neural network that takes words from a vocabulary as input and embeds them as vectors into a lower dimensional space, Embedding Layer usually refers to the weights of the first layer (5). This technique enables us to compute relation among words by using the distance between two embedding vectors. In this thesis we develop both former and new semantical representation of tweets as feature vectors and study the predictive behavior of second hashtags of tweets in time, by comparing different machine learning methods.

Now a day demands on Stream Data processing has been increased, applying machine learning methods on stream of contextual information, can be even more complicated, regardless of its time performance, on the one hand we don't have a fix number of features on the other hand features value may not stay constant in time. High volume of data with time varying data distribution can be seen in this view that subjects and features will not have identical importance and popularity for users all the time and this can be an issue in feature representation tasks. To solve this problem, we analyze the second hashtags time-varying patterns and prediction status in different time frames.

## **1.2 Original Contribution**

The main contribution of this thesis is to extract properly useful features of small documents like tweets. The first phase starts with data cleaning and preprocessing of the tweets included many steps and approaches, important fact that we should consider in this step is meanwhile the elimination of redundant and useless terms in tweets rather taking care of words which contain useful information for the next step prediction tasks. For this purpose, many Information Retrieval and NLP techniques like bag-of-words, NLTK, Gensim, BeautifulSoup have been used, after wards we use approaches like TF-IDF and CountVectorizer in order to get the final numerical feature vectors as standard representation of input data to be used in Machine Learning.

Second phase of this thesis is related to Machine Learning trying to predict the second hash-tags for any of these small documents (tweets). The main issues can be the small number of features in each tweet which makes prediction task much more

difficult, also huge number of predicted values (second hashtags) which can definitely reduce the precision of the final results, make low frequency in prediction. As a remedy understanding the semantic and syntactic relation among these features can be helpful bringing more useful information about features and reducing the dimensionality of the output predicted values by finding similarities among second hashtags. Dimension reduction techniques and Embedding methods can be very helpful solving this issue (6), methods which use multilayer neural networks can efficiently provide a solution by getting deep into semantical layers of terms in a context and by representing them in higher dimension of the problem.

Next is to apply Machine Learning algorithms and try to predict the second hash-tags based on the standard procedure we followed in ML prediction tasks starting from the training data to fit the model and learn how to estimate the prediction values for unseen testing data. However, in this case with considering the fact that we are facing incoming stream data and this way we might face with much time precise predictive behavior in data, like the case where we have almost similar input feature vector but different second hash-tags or output value in Machine Learning in different time. So this thesis tries to show this fact and the solution is on selecting different time windows and wants to show the effect of different window lengths on the ML tasks accuracy on stream data.

## 1.3 Structure of the thesis

This thesis is organized as follows:

Our thesis starts with a dissertation on the state of the art (chapter 2) of the various topics which are involved in our research, such as the importance of Preprocessing and Data Cleaning and techniques to achieve this goal, introduction to the concepts of text processing, Feature Engineering and its steps, Machine learning science and several algorithms related to the work,

The next chapter 3, "Problem Setting", will be devoted to an analysis of the context of the problem and the ontology of the data at our disposal; it will explain in detail the importance time precise prediction in our particular case study

In chapter 4 "Problem Solving". We will provide an overview of the architecture of the system, of its requirements in terms of input data, and algorithms and methodologies to follow and the logic behind our implementation choices.

The "Implementation experience" will be the central point of chapter 5, where we will provide some examples regarding the steps of thesis and provide some codes to get involve in implementation state

Results will be discussed in chapter 6 "Testing and Evaluation", which will also include some description about a baseline and the comparisons between the two methods

The concluding chapter 7 will then recap our experience in researching and creating this application, meanwhile suggesting possible improvements, integrations or future developments to be made on it in order to take advantage of its virtues or to put remedy to some of its limitations.

# Chapter 2

## 2. State of Art

Massive volume of data spread and stored in world wide web and different data centers with high speed every moment, one can manage this stream of data in much more intelligent way to be able to extract latent and valuable knowledge hidden in this quantity of data. It cannot be an easy task many different approaches and science should be involved to provide a method which is quick, accurate and adequate to fulfill this goal which can be have wide range of applications in different sciences. The very beginning issue is to how to deal with raw data and how to efficiently prepare it for machine intelligence techniques to handle in more accurate way. In the following we briefly explain about the theoretical and practical aspects of a procedure to work on a stream on-line data and extract the desired knowledge. First we need to explain about the importance of the preprocessing step on the given data in this thesis. This step was done in 3 steps in this thesis **Extraction:** extracting features from “raw” textual data by TF-IDF,Counter-Vectorizer methods, **Transformation:** Scaling, converting features Tokenizer ,Stop-word Remover. **Selection:** Selecting a subset from a larger set of features.

### 2.1Data Preprocessing

Usually raw data collections from real word have many annoying deficiencies and noise to be processed in any further Machine Intelligence processes, they might have inconsistency problems in names or to be incomplete, with lack of certain attribute values or even noisy with outliers and errors! These are the very first problems we

will face in Machine Learning and Preprocessing is the initial technique to resolve them (7).

Generally Preprocessing of text data is about changing the original text document into a structure with the most meaningful features that enable distinguishing different between text-categories, which is called data mining ready structure. In other words, the process of preparing a document to step into an information retrieval system, means that it should maintain good retrieval performance (precision and recall) (8). It is a crucial step which requires completed process leading to the representation of each document by a selected set of terms. Main goal in Preprocessing is to gain the key words or key features from text and to boost correlation between word and document and the correlation between word and predicted values in Machine Learning. There are different tasks in data Preprocessing, based on the data we have given one might skip or modify one or more steps (9).

**Cleaning:** the main objective of the primary step is dealing with missing values and noisy data, identify outliers, and resolve inconsistencies. Missing values in datasets happens when no data value has been stored for a variable in data samples these missing data can cause further problem in ML and statistical analysis of the data, there can be many strategies how to fix them in data preprocessing as well as,

- Ignoring: for numerical features where class label is missed.
- Replace a mean value to missing values (or majority nominal value)
- Use the mean value for all cases belonging to the same class.
- Use ML methods to predict the missing value

sometimes there are several noisy data in the data set consist of different type of errors and outliers in order to detect these anomalies and smooth or clean the data the following methods are presented however sometimes we might use experts or historical knowledge of data to resolve some conflicts.

- Reorganizing the attribute values and partition them into bins
- Smoothing the noise by bin median, means or boundaries.
- Clustering: grouping attributes next detect and remove outliers
- Regression: By functions with different repressors smooth the noise in data

**Integration and Transformation:** integrate different data files from several databases then apply normalization and compression techniques.

Normalization means to check if all numeric attribute values are in their standard range, it is possible to check their scale to be in specific range for example to transform  $V$  in  $[\min, \max]$  to  $V'$  in  $[0,1]$  the following transformation should be applying:

$$V'=(V-\text{Min})/(\text{Max}-\text{Min})$$

Also in case when min and Max are not known scaling is possible by use of Standard Deviation and Mean:

$$V'=(V-\text{Mean})/\text{StDev}$$

Then we need to Aggregate all numerical or quantitative attributes and Generalize the qualitative attribute values and finally replacing reconstructed values by existed attributes.

**Reduction:** removing irrelevant attributes, reducing number of samples and reducing data mass into a lower dimensional space with considering the consistency of the contexts to have the same analytical results. Many techniques have been proposed like reducing the number of attributes by grouping them into intervals and clusters, or replacing quantitative attributes with qualitative ones (Data Discretization) (6).

### *2.1.1 Text documents Preprocessing*

Many of the frequent words in any languages are often considered irrelevant in Information Retrieval (IR) and text mining. These words are called 'Stop words' (i.e. pronouns, prepositions). Stop words in English language are about 400-500, examples of such words include 'the', 'of', 'and', 'to'. Removing these non-informative words from text can be considered as the first step in text preprocessing, there are many functions providing list of Stop Words in different languages.

Stemming techniques are also used to find out the root/base form of a word. Stemming turn words into their root form, which cover lots of language-dependent linguistic knowledge (7).

Strong hypothesis behind Stemming is that same stem form of different words can represent the same close concept in text so we can replace different word families with their equivalent stem. For example, the words, user, users, used, using all can be stemmed to the word 'USE'. There are many algorithms for stemming the most popular one in English is Porter Stemmer algorithm. Working in this way that a consonant will be denoted by c, a vowel by v. A set of consonant like ccc... of length greater than 0 will be denoted by C, and a set vowels vvv... of length greater than 0 will be denoted by V. Following the basic rule in English language a word has one of the four forms:

CVCV ... C

CVCV ... V

VCVC ... C

VCVC ... V

Document Indexing is another step in Text Preprocessing in order to enhance the efficiency we can use a selected list of words from a document to be applied for indexing the document. Document Indexing consists of choosing the efficient set of



terms based on the whole corpus of documents, and devoting numerical weights to those terms for each particular document, thus transforming each document in to a vector of keyword weights. The weight normally is related to the frequency of occurrence of the term in the document and the number of documents that use that term.

### 2.1.1 Co-occurrence

Co-occurrence analysis has been widely studied in many researches focusing mainly on the domain of Text Analysis, Text Mining and Content Analysis in general. In fact, it is coincidence or more generally the frequency of occurrence of two words from a corpus considering their orders. Mainly it aims at pairwise meaning similarities among words also clarify similarity of semantic within word patterns (10).

In fact, co-occurrence is a primary method to find out semantic correlation among words. Based on the distribution theory similar words appear in similar documents means that they co-occur with similar other words. Accordingly, rather than applying co-occurrence of two words as the scale of their similarities one can have a comparison with this technique among the words with all other words in the document by defining Co-occurrence Distribution for each term as the distribution of average weight of the term in all the documents where that word appears, this concept opens more doors to the “semantic similarity” of two terms. The co-occurrence distribution of a word can also be compared with the word distribution of a text. This definition gives us a measure to determine the importance of a word in a text and therefore can be helpful to include co-occurrence information to identify relevance of keywords with a text (11).

In particular, with aim of statistical tools, the theory of meaning is a mandatory in accounting for two main processes, the method to organize the terms and texts (raw data) into matrix form and an approach in order to represent the outputs in any of the table or graph shape. In other words, it is about the process of converting words into numbers and tables and/or the graphs and vice versa.

At the beginning we need to get deep in representing raw data into matrix form as vector-space modeling, so that each document as a data sample is in a vectorial form of all terms co-occurrence within that context. Also each word here is represented as a vector of all documents which it occurred there.

Subsequently to test terms behavior in “semantic space” we need to take into account stemming and definition of words, if  $WC = [w_1, w_2, \dots, w_n] \in C$  denotes the co-occurrence (WC) of two or more words ( $W_i$ ) within the same context (C), an operational approach need to know exact description of the terms as “word” and “contest”. So that one should clarify how he represent the words. As an example if there are different forms of an ordinary verb (e.g. “show”, “shows”, “showed”, “showing”) whether these terms are encoded as four different terms or just as one stem word of those (e.g. as the root term “show”)

Consider in an analysis problem we have a data matrix that each row-vector is a

document or context unit ( $c_1, c_2, \dots, c_n$ ) and each column-vector represents a different word ( $w_1, w_2, \dots, w_n$ ) Representation of data in each document is encoded in a binary format where ("1") indicates word presence and the absence is represented by ("0") (12). We can obtain a representation as in Fig. 2

By a simple conversion equivalent Square Matrix is obtained which is transformation of the primary data (Fig. 4) here the words are row and column headings while each cell contains the number of context units in which the word  $W_i$  co-occurs with the word  $W_j$ ; but, to illustrate the argument, we can refer to the representation in Fig. 3

|       | $w_1$ | $w_2$ | $w_3$ | ...  | $w_m$ |
|-------|-------|-------|-------|------|-------|
| $c_1$ | 0     | 1     | 1     | ...  | 0     |
| $c_2$ | 1     | 1     | 0     | ...  | 1     |
| $c_3$ | 1     | 0     | 1     | ...  | 0     |
| ....  | ....  | ....  | ....  | .... | ....  |
| $c_n$ | 0     | 1     | 0     | ...  | 1     |

2

Figure 2 rectangular matrix

|       | $w_1$ | $w_2$ | $w_3$ | .... | $w_m$ |
|-------|-------|-------|-------|------|-------|
| $w_1$ |       | 5     | 13    | .... | 8     |
| $w_2$ | 5     |       | 2     | .... | 11    |
| $w_3$ | 13    | 2     |       | .... | 6     |
| ....  | ....  | ....  | ....  | .... | ....  |
| $w_m$ | 8     | 11    | 6     | .... |       |

Figure 3 square matrix

<sup>2</sup> www.soc.ucsb.edu

|       |        |       |          |        |       |       |        |       |        |
|-------|--------|-------|----------|--------|-------|-------|--------|-------|--------|
| $w_1$ | "book" | $w_3$ | "Harris" | "late" | $w_6$ | $w_7$ | "read" | "you" | "work" |
| 0     | 1      | 0     | 1        | 1      | 0     | 0     | 1      | 1     | 1      |

*Figure 4 representation of an utterance*

The question is: using a similar representation, in which each word has a sole attribute (i.e. it is, together with other words, an "element" of the same set), which kind of meaning can we extract and analyze? Or rather: what kind of meaning can we infer? Also given that, in order to "extract" meaningful patterns, co-occurrence analysis requires comparison.

## 2.2 Feature Engineering

In machine Learning Features are measurable property of the input data, they are usually numerical but we can also use qualitative or structural form of features such as strings or graphs. It is crucial to select intuitive and informative features of data to improve the efficiency and performance of Machine Learning. Usually to collect all features of the input we might face a huge and redundant set of raw features which need to be organized in many aspects so that be ready for ML and Pattern Recognition applications and will aid in learning process. A set of numerical features is called Feature Vector this numerical representation of objects will facilitate many statistical and mathematical methods in ML (13).

Feature Engineering is a vital process to ML, which is about using knowledge of raw data and transform it to the features that can represent data more efficiently to predictive models so that the better preparation of features the better the model result with unseen data.

Actually Machine Learning in general will learn the solution in the training phase and with the sampling data (14). We need Feature Engineering to find the best representation of sampling data to efficiently help ML algorithm learning the solution in your problem case.

To clarify this approach, imagine a text categorization problem where the task is to train a model for classifying a given document as spam and not spam. If we represent a document as a bag of words (unigrams), the feature space consists of a vocabulary of all unique words present in all the documents in the training set. For a collection of 100,000 to 1,000,000 documents, we can easily expect hundreds of thousands of features. If we further extend this document model to include all possible bigrams and trigrams, we could easily get over a million feature which is not an ideal case because high ratio of these features are not really useful and unnecessarily will increase the complexity.

To track process of Feature Engineering one needs to follow the ML process as well.

A picture relevant to our discussion on feature engineering is the front-middle of this process.

1. **Data Collection:** Integrate data, collect it together.
2. **Data Preprocessing:** give a Format to data, clean it from outliers.
3. **Data Transformation:** where Feature Engineer happens.
4. **Model Evaluation:** find a model and optimize it.

Feature engineering can be an iterative method until the time that these features are suitably fits into the selected model the process can be seen as following:

**Brainstorm features:** based on the problem, search for all possible features

**Devise features:** apply feature extraction and construction

**Select features:** feature selection based on the scoring process

**Evaluate models:** Estimate model accuracy on unseen data using the chosen features.

## 2.2.1 Features Extraction

In Machine Learning we cannot feed the algorithms with raw data. Data might consist of different quantitative and qualitative features but Machine Learning algorithms have mathematical or statistical background so a transformation is needed to create an acceptable data format out of the original data. Usually data is consisting of fixed number of features (input variable or attributes) with different types, binary, continuous or categorical. So finding good data representation is important task this representation format is called Feature Vector, a numerical representation of features of each data sample (15).

Converting “raw” data into a set of informative and non-redundant features sometimes need some human expertise, also its construction can be developed by several automatic methods or sometimes it can be integrated during model processing (e.g. the “hidden units” of artificial neural networks) (15). Sometimes feature construction has been accomplished at Preprocessing phase. To explain the steps first we need to introduce some notations. Let  $x$  be a pattern vector of dimension  $n$ ,  $x = [x_1, x_2, \dots, x_n]$ . The components  $x_i$  of this vector are the original features. We call  $x_0$  a vector of transformed features of dimension  $n_0$

For example, consider, a feature vector like  $x = [x_1, x_2]$  where  $x_1$  is a height in centimeters and  $x_2$  is a width in meters. In order to enable these features for further comparison and mathematical computations we need to normalize them by scaling and centralizing the data as the following:

*Equation 1 feature normalizing*

$$x_{0i} = \frac{x_i - \mu_i}{\sigma_i}$$

Where  $\mu_i$  and  $\sigma_i$  are the mean and the standard deviation of feature  $x_i$  over training examples.

While constructing features different cases might show up. For example, to deal with structured or ordinal data there are convolution methods which are trying to manipulate or encode the data into an informative feature. Also high dimensional data can be projected into lower dimension data by maintaining as much information as possible. Well-known approaches are Principal Component Analysis (PCA) and Multidimensional Scaling (MDS) (Kruskal and Wish, 1978).

Feature extraction requires elimination in the number of assets in data representation this can be important because great number of features need huge amount of memory storage and will slow down the computation speed, so while working with completed huge datasets the key point is to prune inessential part of data and find informative and key variables of data. Poor Feature Extraction can cause further problems in classification tasks like “over fitting” and further inefficient generalization problem with test data.

In particular, we should be careful about not losing information at the feature construction stage while projecting and eliminating the features. It may be a good idea to add the raw features to the preprocessed data or at least to compare the performances obtained with either representation. It has been recommended that missing on the side of being very comprehensive worth risking in useful information elimination.

### **2.2.2 Features Selection**

Feature selection is the process of selecting subset of features which are the most relevant with the training and generalization tasks in ML. For example, in Text Categorization, effect of feature set size on the performance is an early concern. Finding optimal subset of variables is NP-hard, usually we need some basic ML information and measurements on feature space  $F_i$  like correlation coefficient and prediction accuracy to be examined on a validation set. Methods usually use a sub-optimal greedy search algorithm (16).

However, Feature Selection is not exactly a Dimension Reduction problem. While in dimensionality reduction the process is based on different projections to create new combinations of features in a dataset, but in feature selection by adding and removing existed variables in the data finalize the feature vectors without changing them.

Different methods are proposed so far; one can be filtering method. Base constraint the univariate of the method which is based on independency of features. By statistical computations, features selection is based on their corresponding scores to be selected or to remove (17). Another method can be Embedding Methods; the most well-known one is generalization methods. With this case only features with greater effect on boosting the accuracy of ML models will be selected. This method is also called Penalization base on the extra concentrations on the optimization of predictive ML algorithms that can bias the model to less coefficients and therefore lower complexity (e.g. LASSO).

Generally, with feature selection we should always consider several principles like, constructing a proper and normalized 'ad hoc features' from the domain knowledge. With independent features we need to develop sets by product of features and constructing conjunctive features. Also based on the predictive methods are used one should use different selection methods, like with linear predictors forward selection with “prob” or zero norm embedding methods should be applied, if you have enough data and computational resources then any backward selection, correlation coefficient or embedding methods can be compared and any linear or non-linear model can be combined with the best selection method.

## 2.3 Machine Learning:

Machine Learning has been considered as a novel knowledge which began almost with the birth of computer science but if we look at past human efforts on make machine learn to do some tasks automatically we will face a great historical background related to this knowledge where at 1642 Blais Pascal designed the first mechanical adding machine just able to add and subtract and mainly consists of wheels and gears, later in 1890 Herman Hollerith create new version of mechanical calculation with further functionalities for huge statistical computations which was quit faster and worked with punch cards. Finally, the first calculation machine with contribution of electronic was built at IBM in 1945 (Mark I), but the first fully electronic computer (ENIAC) in 1946 was the turning point. So far people were trying to build aromatic machines able to process high level computations much faster than humans. But later they go further trying to make machine learn several functionalities and train it for decision making problems.

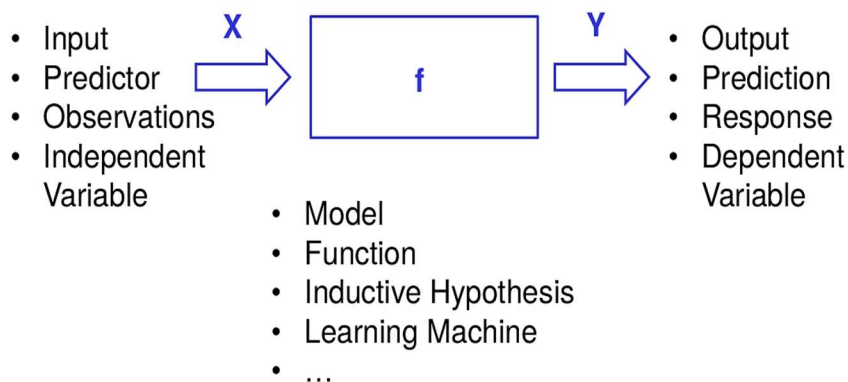
In 1952 Arthur Samuel used the game of checkers to create the very first learning program in IBM. The algorithm behind it was kind of 'Supervised learning' which observed which acts were wining strategies and adapted its programing to merge those strategies. In 1990's Machine Learning has wide range of applications in Text Learning, Adaptive application, Data Mining, web applications and language learning and further progress continued on Supervised, Unsupervised and Reinforced Learning methods it also brought explosion in Adaptive programing which cause advanced in optimization problems and enhancing the accuracy of process.

Machine learning apply different computer algorithms for learning from data to do several tasks, these tasks can be making decisions, completing tasks, make prediction based on available information or to behave intelligently. In fact, Machine Learning (ML) is studding deeply some amount of data or observations and learn to act better in the later with unseen data according to what was experienced before. These data can be such as samples, instructions or direct experiences (18). The main goal and overall emphasis of machine learning is to do the learning task automatically without human intervention and we rely on computer abilities to learn the learning algorithm by its own and only based on training process.

ML has a very close relation to statistics so that as like statistical learning methods. it is trying to build a model from input data samples and in both cases the goal is to learn from this model given data to find the predictive behavior of data, but the difference is that in ML after building the model we have data-driven predicted values rather than following a strictly statistical instruction to produce the output (19). Therefore, they are both same consents but they are represented in two different ways, with the same goal.

Building model out of data cannot be a clear consent in the first step however it is a very important part in ML. Data model represents relationship between data variables in order to predict the output, this model can as any type of parametric or non-parametric function but in general there is a constant terminology in all problem cases that the observed data are used to build the model and the output of this

model given the observed data is demanded predicted value:



3

Figure 5 overview of fitting a model in ML

Representing two main categorization of methods in ML can be useful for better understanding the machine learning approach, the first categorization can be based on given tasks which can be, Supervised Learning, Unsupervised Learning and Reinforcement Learning (18).

Supervised learning methods are involved in building a model for predicting, or estimating, an output based on one or more inputs. The input data is called training data which is a pair of data and its corresponding known label, supervised learning problem can be formulated as follows.

Let  $\{(x_i, y_i)\}_{i=1}^N$  be a set of  $N$  training examples. Each example can be considered as a pair of  $(x_i, y_i)$ , where  $x_i = \langle x_{i,1}, x_{i,2}, \dots, x_{i,T_i} \rangle$  and  $y_i = \langle y_{i,1}, y_{i,2}, \dots, y_{i,T_i} \rangle$ . As an example in part-of-speech-tagging one pair in the form of  $(x_i, y_i)$  might consist of  $x_i = \langle \text{doyouwantfrieswiththa} \rangle$  and  $y_i = \langle \text{verbpronounverbnounprepronoun} \rangle$ .

The aim is to construct a classifier  $h$  that can estimate the predicted value of a new unseen label sequence  $y = h(x)$

Training pairs can be considered as cases when our input data  $x_i$  is an email and the corresponding label  $y_i$  is either spam or not-spam. The model further will be built through a training process, where we select a type of model among wide range of available models and then by feeding the training data to the candidate model and predicting the output using training data and the model we can tune the model and correct the model when these predicted values are actually wrong. This iterative training process continues until the candidate model achieves a desired level of accuracy on the training data. Problems of this nature occur in fields as diverse as business, medicine, astrophysics, and public policy.

For example, a Supervised learning with emails as input and two outputs or response

<sup>3</sup> ISLR



to any input as Spam or Non-spam. First we are given bunch of email contents that have been “tagged” for some feature, like spam/non-spam (classifiers). Use these email documents as a training set that produces a statistical model. Apply this model to new email. Retrain model on larger/better dataset to get improved results. This sort of Machine Learning approach has several useful properties that make it possible to easily measure the error (because we have known all possible predicted value) enhance the accuracy of the model.

In contrast, Unsupervised learning describes the somewhat more challenging situation in which for every observation  $i = 1, \dots, n$ , we observe a vector of measurements  $x_i$  but no associated label  $y_i$ . In this setting, we are in some sense working blind; the situation is referred to as unsupervised because we lack a response variable that can supervise our analysis. The main challenge is that the approach here tends to be more subjective, and there is no simple goal for the analysis, such as prediction of a output  $y_i$ . Unsupervised learning is often performed as part of a heuristic data analysis. Also it is not easy or even possible to check the outcomes obtained by this methods, since there is no general mechanism for validating the results.

Imagine the case when we have  $N$  observations like  $X = (x_1, x_2, \dots, x_N)$  and we want to estimate corresponding  $y_i$  for each  $x_i$  in the observation data without any supervisor or degree-of-error. In Unsupervised learning the dimension of  $X$  is usually much higher than in supervised learning. The main difference is that here we have no prior information about output in the training set so the output is estimated based on similarities among several properties of the input data. This lack of knowledge about exact features of the output or response value make further problem in checking the correctness of the model.

Reinforcement Learning is the third task in this category in ML which is also a branch of Artificial Intelligence, can be defined as learning how to map situations to decisions or actions so as to maximize the rewards. In this case we have an agent who is interacting with surroundings and has a goal, both the agent and the environment have several states, according to the state which the agent can perceive, he will choose an action the learning section is refers to the signal award dedicated to the agent based on the selected action, agent can adapt his behavior by time based on the feedback on his behavior this reward feedback is known as the reinforcement signal. Reinforcement learning can be seen as a kind of problem for which there are many different algorithms and solutions.

To recall in machine learning we have set of input data  $X = (x_1, x_2, \dots, x_N)$  where each  $x_i$  can be a vector of features of its data sample, we have a model that its task is to estimate the output corresponding to each  $x_i$  as  $y_i$ . The other popular categorization of machine learning methods can be seen from the view of output type of the problem. We have 5 main classes; Classification, Clustering, Regression, Density Estimation, Dimension Reduction.

Classification is one of the most well-known techniques in machine learning that has been widely used in different fields like semantic analysis, risk assessment, computer

vision and many other fields. The main goal or the main difference of classification with other methods is that here the predicted value is a qualitative and discrete which is called classy from some input  $x$ . this topic will be explained more in detail in the next section.

Although if the predicted value is quantitative and continuous then the case is a regression problem. Generally, we assume regression methods as a subset of supervised Learning methods, because the standard regression models can be used in the machine learning framework to learn from the training data and provide outcome predictions based on the inputs. Several example of supervised learning methods, that apply regression-based methods can be generalized linear models, logic regression, and regularized regression and tree-based methods like decision trees and random forests.

Density estimation is an unsupervised learning method in machine learning which is trying to learn the correlation among features of a data. In fact, each data sample is considered as a vector of corresponding features of that input data (20). These attributes or features are random variables which can be any of types of continues or discrete. When we talk about random variables, probability theory always takes a part. So in order to learn from data we need to learn the probability density of the features:

**Data:**  $D = \{D_1, D_2, \dots, D_n\}$   
 $D_i = x_i$  a vector of attribute values

**Attributes:**

- modeled by random variables  $X = \{X_1, X_2, \dots, X_d\}$  with **continuous or discrete valued variables**

**Density estimation attempts to learn the underlying probability distribution:**  $p(X) = p(X_1, X_2, \dots, X_d)$

With considering two important facts that these samples are from same distribution but independent from each other, the goal actually is to compute the probability distribution over variables  $X_i$ ,  $P(X)$  using examples in  $D$  (18).

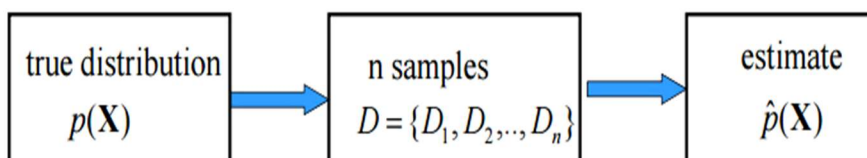


Figure 6 probability distribution estimation

In many problems, we are facing high-dimensional data vectors but for many reasons like computational complexity or achieving higher accuracy, we try to make the data lie near a lower-dimensional manifold. In fact, high-dimensional data are multi fold, complex evaluation of an underlying source, which cannot be directly computed. Projecting the data into a lower dimension problem can aim on learning a suitable low-dimensional manifold from high-dimensional data which is equivalent to learn this underlying source.

Dimensionality reduction can be defined as the process of deriving a set of degrees of freedom which can be used to reproduce most of the variability of a data set. Consider a set of images produced by the rotation of a face through different angles. Clearly only one degree of freedom is being altered, and thus the images lie along a continuous one-dimensional curve through image space.

Regardless of different categorizations in ML we face several general difficulties in all cases. Model in ML can be a mathematical or statistical function  $F(X)$  which is supposed to receive the input data and estimate the output value  $Y$ . There are wide ranges of functions from linear or non-linear, Parametric or non-parametric and etc. Choosing proper model needs vast information on the distribution and behavior of data. In fact, this function is helping us to model the data and to learn from the data, so it is obvious that we cannot estimate these predicted values with no error! Reducing this error and increasing the accuracy can be seen as another important issue in this regards. However by concentration on the model and increasing some factors like degree of freedom or reducing the training error one can increase the accuracy on training data where new problem will be Bias-Variance trade-of which in summary happens when you are trying to reduce the training error and learning the data as much as possible by enhancing accuracy of the model on training data (increasing degree of freedom and consequently increasing the bias of the model) but on the other hand the final test error is not decreasing as well which means we are not really acting well on the unseen data. Understanding and coping with such tasks is not really easy and need enough knowledge and experience in this regard.

### 2.3.1 Classification

In Many cases, the output is qualitative. For example, eye color is qualitative or categorical, taking on values like black, blue, brown, or green. In this chapter, we study approaches for qualitative responses a process known as classification which is generally the main ML method used in this thesis. Predicting a categorical output for a observed data can be referred to as classifying that observation, since it involves allocate the input data to a class (17). Strategy for methods in classification problems is to first predict the probability of each of the classes for all of the output variables as basis for making the classification, so far it is something similar to regression.

Also in classification setting we have a set of training data  $(x_1, y_1), \dots, (x_2, y_2)$  that are used to make a model which here is called a classifier. The main goal of the classifier is performing perfectly both on the training data also on the test set which are not applied in the train set.

In the following we would like to give a high level example of one of the classification methods in a simple case. Instead of modeling directly the categorical outputs Y a method name logistic regression can model the probability that Y belongs to a specific class label. We need to model the relationship between  $P(X)=\Pr(Y=1 | X)$  and X, in linear case we have the following:

*Equation 2 Linear function*

$$P(X) = \beta_0 + \beta_1 X$$

we also have a function for modeling P(X) which is called logistic regression and is as the following:

*Equation 3 logistic regression*

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

by playing with this form and taking log of it the other form of logistic regression can be the following:

*Equation 4 log function*

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

in order to fit or learn this model we should first estimate the unknown parameters  $\beta_0, \beta_1$  based on the training set. For linear coefficients least square approach can be used but in nonlinear models Maximum Likelihood is a very common approach for learning the models. In fact, also least squares can be considered as a special case of maximum likelihood, the formula is the following:

*Equation 5 Maximum Likelihood*

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

Only those estimates of  $\beta_0, \beta_1$  are chosen to maximize this likelihood function.

### 2.3.2 Text Classification

The main subject area in classification here in this thesis is focused on text data, this type of problems is usually known as Text Classification. Here we are given an exposition of a document  $d \in X$  where  $X$  can be considered as a document space which is a high dimensional space, plus a fixed set of human-defined categories or classes. Each of the documents can be in multiple, only one or no class at all:

$$C = \{c_1, c_2, \dots, c_j\}$$

For instance, China and documents that talk about multi-core computer chips above. We are given a *training data observation set*  $D$  so that in this set we have pair of each document and corresponding label of it,  $w\langle d, c \rangle$  here  $\langle d, c \rangle \in X \times C$ . For example:

$$\langle d, c \rangle = \langle \text{Paris joins the World Trade Organization}, \text{France} \rangle$$

for the one-sentence document Paris joins the World Trade Organization and the class France.

As before the main goal is to learn from this method so we are seeking for a learning process to learn the classifier or *classification function* that maps text documents to categories:

*Equation 6 classification method*

$$\gamma: X \rightarrow C$$

Obviously it is a supervised learning because a supervisor (the human-defined classes and labels training documents) exists to address the learning process in order to prevent categories overlap each of the classes are treated as a separated binary classification. We denote supervised learning function by  $\Gamma$  and write  $\Gamma(D) = \gamma$ .

As before this learning model  $\Gamma$  receive an input training set as  $D$  to estimate and to learn the classification method  $\gamma$ , the first step in Text classification is to convert the documents which are strings of characters into a suitable representation for the learning algorithms (20). In text classification terms or words are taken as features, these features are internal to the classification function and there are many approaches to modify these features to a desired format or even to build new latent features variable these methods and processes are referred to Feature Engineering, so far feature engineering was mainly done by human rather than a process done by Machine Learning but now a days different factorization methods are presented which can wisely extract useful latent features from text (21). Acting good enough on feature engineering will cause notable improvement in final Classification performance.

Here words are brought together in sets and groups with similar effects on the classification task. For example, dates, chemical formulas or Identification numbers like ISBNs. Using these terms straightly in vocabulary set will increase greatly the size of the set although they do not really convey any information or power in classification function (22). So to reduce the size of features and maximize their

efficiency one should transform these terms in to an equivalent subset like regular expressions rather than accuracy enhancement this way we can also speed up the classification task.

From Information Retrieval research it has been cleared that words root (stems) can work in advance for representing the document terms so that their order in the document even not that important. Considering this important fact in IR we will find an efficient feature representation for text. In this representation model we store features in text which are words in the document  $asw_i$ , with the number of times that the word appears in that document which is the value of each feature (word). As explained before unnecessary features like "stop words" or infrequent terms will be discard from the feature space. So even with this presentation we might have a very high feature dimensional space with 10000 features or more so feature selection is needed. Scaling the dimensions of the feature vectors has been affirmed by IR to the *inverse document frequency* "IDF".

One of the most endorsed representation methods for text classification is *bag-of-words* (23). When we have a set of documents and by using *bag-of-words* (BoW) we want to create the feature vectors, first model learns a vocabulary set from all the documents, then models each document by the words frequency or counting the number of times each word appears in that document. For example, consider the following two sentences:

**Sentence 1:** *"the green pot is on the table"*

**Sentence 2:** *"the table and the green pot are in the room"*

From these two sentences, our vocabulary is as follows:

{the, green, pot, is, on, table, and, are, in, room}

To get our bags of words, we count the number of times each term appear in each sentence. In Sentence 1, "the" appears twice, and "green", "pot", "is", "on", "table" each appear once, so the feature vector for Sentence 1 is {the, green, pot, is, on, table, and, are, in, room} which by considering number of words occurrence in the sentence we have features for each sentence as the following:

**Sentence 1:** {2, 1, 1, 1, 1, 1, 0, 0, 0, 0}

**Sentence 2:** {3, 1, 1, 0, 0, 1, 1, 1, 1, 1}

in many data sets we have a lot of data samples like sentences here, which will cause a large vocabulary. To limit the size of the feature vectors, we should choose some maximum vocabulary size. (remembering that stop words have already been removed)

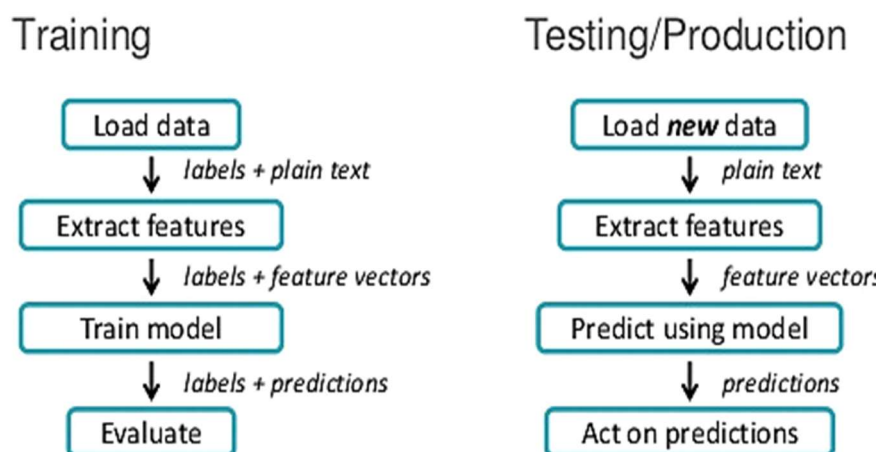
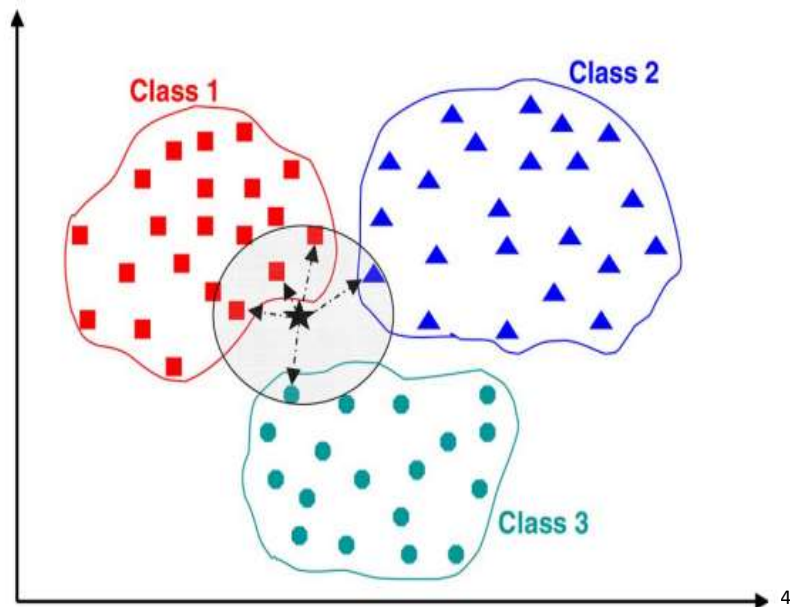


Figure 7 Testing & Training in ML

Last approaches on text classifiers is focused mainly on only a set of categorized training samples that are classified manually and has lower cost to be produced. By learning from a pre-classified training set a classifier can be designed. This means that in supervised learning a classified set is needed in the training set which can be perceived as a disadvantage because the ML final precision is depends on the the state efficiency of the data samples.

Totally there are two main type of text categorization proposed so far, ML based or Rule based techniques, in rule based approaches the document is classified based on already manually designed rules. Difference in ML approach is that these facts and comparisons is mainly based on the classified documents which can build automatic rules, here we have a lower precision than rule based approaches. Different examples of these approaches are described in the following.

K. Nearest Neighbor (KNN) is a non-parametric algorithm (does not make any assumption on the distribution of the date) used for both classification and regression. In many cases, it happens that the data is not submitting any scientific hypothesis on its distribution like when we have Gaussian mixture for this cases non-parametric methods like KNN can be useful this method does not need any generalization from the training set which means that it doesn't need to have test set all learning and testing processes are applied in one step and only on one training input sample so in these type of method learning process and the final output or decision is based on the training set(or subset of that) (24). KNN assumes data in a metric feature space for this reason data can have the concept of distance in such space (24). It works in a way that a label of data sample is set with regards to the most common label among k nearest training data (with the smallest distance from the data subject). This Method Perform well on multi-class text classification assignments although from time-consumption point of view when we have a large set of training documents it will take more time to be proceed with KNN.



*Figure 8 KNN classification method*

Naïve Bayes classifier is a supervised approach using the fundamental concepts in probabilistic theory and statistics, a very powerful method based on Baye's theory. It can solve predictive problems and is able to measure uncertainty about the model based on the probability distribution of the output values. Posterior Probability of each input textual samples belonging to specific category is calculated based on prior probability of that class labels and the likelihood of the data (measured data) so that it assigns document to the class with the highest posterior probability. A very strong assumption in this model is based on independence of features of the document.

<sup>4</sup> [www.ieeeexplore.ieee.org/stamp/stamp.jsp?arnumber=4342786](http://www.ieeeexplore.ieee.org/stamp/stamp.jsp?arnumber=4342786)



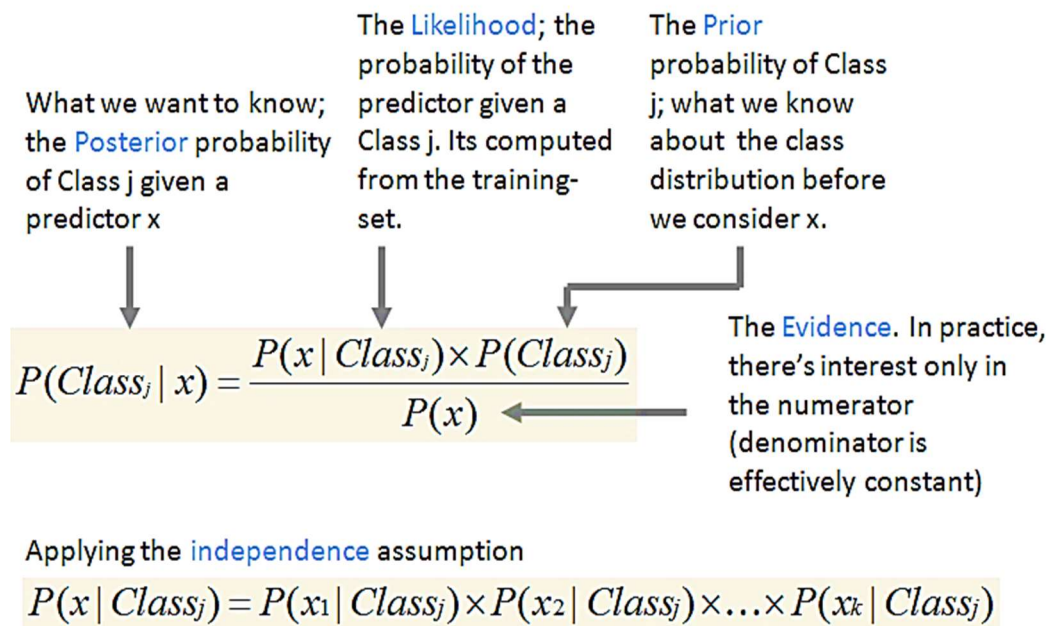


Figure 9 Naive Bayesian Method

A Support Vector Machine is another supervised classification algorithm that has been extensively and successfully used for text classification task. Method define decision boundaries based on the concept of decision planes that. A decision plane is one that separates between a set of objects having different class memberships. As in text categorization we usually have a high dimensional input space which means that we have huge number of features which are corresponding informative terms in the document, but learning from this complex data space is not easy and we need to deal with it in order to find a proper hyperplane and be careful about bias-variance trade-of to prevent over fitting. In SVM we need to build a hyperplane between data samples from different classes, by mapping the problem into a higher dimensional space we can find the even non-linear hyperplane between different classes of data. Most text classification problems are linearly separable: All categories are linearly separable and it declares that the SVM hyperplane should be linear in text classification. In the Following image<sup>5</sup> it clarifies how the method find a proper hyperplane by mapping input data to a higher dimension feature space:

5 .<http://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html>

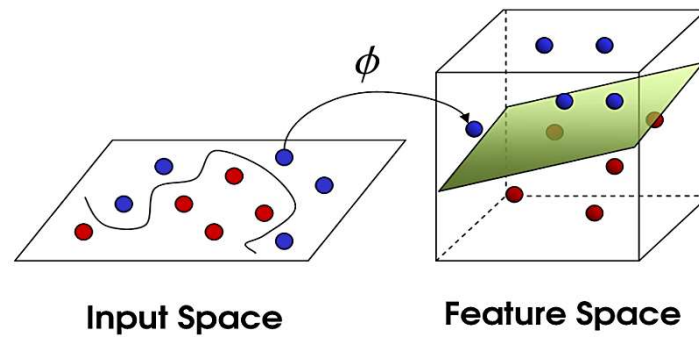


Figure 10 Support Vector Machine

### 2.3.2.1 Short-text Classification

With the boom of e-commerce and social media, short texts, such as instant messages, micro-blogs and product reviews, become more available in diverse forms than before. These short forms of documents have become convenient presentations of information (25). It is becoming more and more important to understand those short text documents and to efficiently detect what users are interested in. Unlike long documents such as news articles and blogs, it is hard to measure similarities among these short texts since they do not share much in common (Phan et al., 2008). This poses a great challenge to short text classification (STC).

The task of short text classification can be described as follows: given a short text  $S$ , the aim is to identify its target theme  $T$ . Several supervised learning approaches have been proposed for short text classification. They have been shown to be effective and yielded good performance. These approaches are effective because they leverage a large body of linguistic knowledge and related corpora. However, the supervised learning approaches depend heavily on manual annotation, which is labor intensive and time consuming, and often suffer from data sparseness.

To tackle the above problems, we exploit word embedding. A word embedding  $W: \text{words} \rightarrow \mathbb{R}^n$  is a distributed representation for a word which is usually learned from a large corpus (26). Many researchers have found that the learned word vectors capture linguistic regularities and collapse similar words into groups (Mikolov et al., 2013b).

In micro-blogging services such as Twitter, the users may become overwhelmed by the raw data. One solution to this problem is the classification of short text messages. As short texts do not provide sufficient word occurrences.

## 2.4 NLP

Natural Language Processing (NLP) is a research field or application in computer science demanding to analyze the way a computer can learn and employ the text or speech to accomplish further assignments and learning processes. By exploiting ML and studying patterns in data NLP have been succeed to enhance many self-understanding programs. In fact, by simulating the between human binges behavior of using and understanding the language, NLP goals can be reached which means that many different sciences fields like psychology, robotics, AI, computer engineering and mathematics can be involved in this area of research.

As a matter of fact, natural language refers to the common languages like English, French, Persian etc. which is hear or spoken by people all over the world for communication, every language can be seen as a mechanism with a collection of laws and tokens where tokens are synthesized and exploit to transfer information, but rules are used to arrange and handle these tokens (8). Natural Language Processing as a subfield of Artificial Intelligence gather all the tools and techniques to make machines understand all textual elements and fundamental in human languages. There are wide range of NLP applications like text processing, multilingual information retrieval, speech recognition, machine translation and artificial intelligence.

General steps in NLP can be 5 as the followings, the first one is to split the text into paragraphs, sentences and terms and determine or investigate all the terms and expressions in that text and related to the language vocabulary, the lexicon is the vocabulary of a language, this step is called Morphological and Lexical Analysis. Second step is Syntactic Analysis which is mainly refer to the grammar of the corresponding language which means by analyzing the words in a sentence from grammatical point of view the machine can detect sentences which are grammatically wrong in that language this can be done based on structure of the terms in that sentence and their orders (27). Next can be Semantic Analysis where we analyze the meaning of the context by mapping the syntactic structures extracted from syntactic analyzer to the terms in the scope of task for example “brightless room” which would not be accepted at this step by the analyzer because bright less does not make any sense (4). Discourse Integration is the 4<sup>th</sup> step where for the meaning investigation we should also consider the sequential sentences in a text which clarifies that the meaning of the current sentence is related to the previous sentences for example pronouns like “they” refers to several objects already pointed in the former sentences. Finally, the last step which is Pragmatic Analysis which is investigating the interpretation the concept of sentences rather than only the meaning of each sentence this way we can drive the purpose of the language in different situations.

## 2.4.1 Word Embedding

we have already explained about different data representation methods which has been used in feature engineering or for text document representations but here after a brief description of NLP we can propose a recent approach more specific in word representation, as a collection of feature learning and language modeling approaches. The process is to reduce the dimensionality of the document features or terms by projecting them into a lower dimensional space then each term can be represented with a scalar vector. Many methods can be used in this regard, supervised or unsupervised approaches like probabilistic methods, neural networks or dimension reduction techniques.

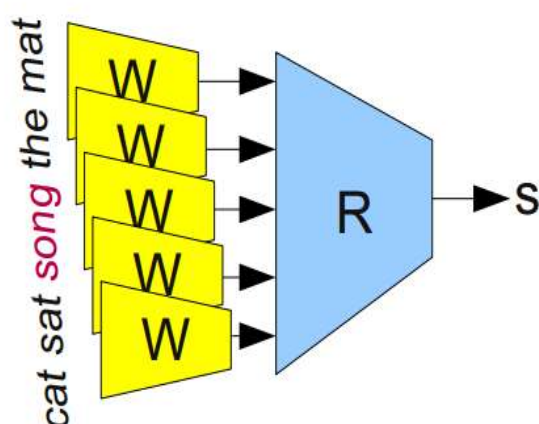
Word embedding is also called distributed word representation, word embedding is dense, low dimensional, and real-valued vectors representing terms of a document. consider an embedded feature space where each dimension is the latent features trying to contain well-formed and syntactical linguistic attributes, in fact this kind of representation is trying to cluster huge number of data and obtain a compact representation of these features. Usually with neural network methods one can obtain a sufficient predictive model (Bengio, 2008) but not so fast adequate to the size of the vocabulary although many approaches presented recently to reduce this linear dependency on vocabulary size and enable processing of very large documents (Collobert & Weston, 2008; Mnih & Hinton, 2009)

as we are mainly involved with language terms or words in NLP so the way of words representation can play an important role and based on what mentioned before researchers get to this point that considering every single word or even selected ones as features symbol is not really an efficient way and rather they understand that considering the similarity and dissimilarity among words. Word embedding found on Harris distribution hypothesis which has been widely used in NLP specially in the word representation area and affirms that terms in similar context have the same meaning. Based on this distributional assumption one can describe a matrix  $M$  to represent word terms in different context so that each row  $i$  corresponds to a word, each column  $j$  to a context in which the word appeared, so each matrix element represents a measurement among a word and a context (28). This way the dimensionality reduction operations can be applied on rows of this matrix were words are placed. Even better representation can be when for every word in this matrix we have a dense vector which can contain more useful syntactical and semantical information of the corresponding word, this recent type of representation can be achieved by models inspired from neural-network modeling which has been perform well in a variety of NLP tasks and referred to as “neural embeddings” or “word embeddings”.

So in fact it can be seen also as a deep learning field of research, a word embedding  $W: \text{words} \rightarrow \mathbb{R}^n$  is a parametric method to map words in some language to high-

dimensional vectors. At first  $W$  is initialized with some random vectors for each word but later it will learn to update its vectors and have a more meaningful vector for further knowledge extraction tasks. While it is clear that the training objective follows the distributional hypothesis – by trying to maximize the dot-product between the vectors of frequently occurring word-context pairs, and minimize it for random word-context pairs. For example, using this approach it is possible to understand whether a sequence of  $n$  words ( $n$ -gram) is valid or not in the following you can see a famous 5-gram example from Bottou (2011):

In this example we have a 5-gram sequence (“cat sat on the mat”) then break part of it replacing one word with a random one like (“cat sat song the mat”), obviously this sentence is no more meaningful.



6

*Figure 11 Modular Network to determine validity of a 5-gram example*

This model will run each word in the 5-gram through  $W$  to get a corresponding vector representation of each word and feed those into another ‘module’ called  $R$  which tries to predict if the 5-gram is ‘valid’ or ‘broken.’ Then, we’d like:

$$R(W(\text{“cat”}), W(\text{“sat”}), W(\text{“on”}), W(\text{“the”}), W(\text{“mat”})) = 1$$

$$R(W(\text{“cat”}), W(\text{“sat”}), W(\text{“song”}), W(\text{“the”}), W(\text{“mat”})) = 0$$

To enhance higher accuracy, we can play with parameters in  $W$  and  $R$ , however we are more interested in learning  $W$ . Generally, this type of words mapping provides lots of discriminative information to us, so that similar words in the word space are close to each other. In other words, we can get to this concept also in the words embedding space which also in that case similar words are closer to each other. It seems natural for a network to make words with similar meanings have similar vectors. If you switch a word for a synonym (eg. “a few people sing well” → “a couple people sing well”), the validity of the sentence doesn’t change.

<sup>6</sup> [www.colah.github.io/posts/2014-07-NLP-RNNs-Representations/](http://www.colah.github.io/posts/2014-07-NLP-RNNs-Representations/)

## 2.5 Neural Networks

Neural network (NN) is a network of simple processing units called neurons which are connected to each other and each processor is able to perform several simple processing tasks. The first line or layer of neurons are exposed to the input data which are received by sensors which are interacting with environment, neurons in other layers receive the data through weighted links among previously activated neurons.

Artificial Neural Networks is known as a ML method which has been widely used and overcome the close competition in several problems among different method of pattern recognition and machine learning like speech recognition and computer vision, which are not easy to solve with previous approaches. What we mean by the learning process in this method is to search for the weights which can make NN to represent the desired tasks which depends on the chains of the computational phases or layers of neurons, an example for such tasks or behavior can be detecting the useful edges in a picture (29). At each computational phase there might be functions either linear or non-linear which have to transform the activation values of the network.

There have been many convolutions in the history of NN so far but in 1960s and 1970s when different approaches with non-linear layers spearheaded a challenging competition started where Back Propagation (BP) as a model designed by means of gradient descent methods for supervised learning on separated networks. Although BP was not really applicable for deep learning so that in deep layers this method was not really practical and smooth in training phase. Finally deep learning was possible in practice with Unsupervised learning. Afterward deep neural networks start a serious competition with many other machine learning specially in 2009 supervised neural networks overcome many other pattern recognition techniques like with kernel methods which drew much more attention to this concept of knowledge. Rather than many achievements from 2011 in computer vision, NN shows a high performance in Reinforcement Learning by unsupervised learning.

To explain more in detail one of the NN models we can consider a simple feed forward NN which has been widely used recently and in this thesis we used this model in word embedding, in the following we also explore the learning process in this NN type. A feed-forward neural network or acyclic NN is another classification algorithm where every neuron in a layer is fully connected with all other neuron units in the prior layer. Each interconnected links might have individual Weight, which make different operation on the transmitted data in the network. Input data enters in the network and transit the layers one by one which at every layer data will pass through a process until it arrives to the output. This can be considered as the learning process where the connection weights in feed-forward neural networks will change in order to assign the highest output value to the right class. When it is performing a normal

classification task there is no feedback in between layers it is the reason why this method is known as acyclic or feed-forward. So for learning you may have several objects that you need to choose, like the patterns list as input, feed-forward neural network as the classifier and the class which can be the correct output (30). The pattern will move forward in layers to get to a specific output layer, each neuron or unit in this last layer shows specific class or category which in fact is the output of the network and will be compared to other ideal cases based on the classified selected pattern (the output value of the other units in the last layer would be smaller than the one with the right class)

Using deep learning for NLP applications has been investigated recently (inter alia Bengio et al., 2003; Henderson, 2003; Collobert & Weston, 2008). In most cases, the inputs to the neural networks are modified to be of equal size either via convolution and max-pooling layers or looking only at a fixed size window around a specific word. many methods which can deal with substance of recursive behavior in natural language so that based on variables in a sentence they can learn the categories of expressions and feature embedding which can obtain its semantical construction more deeply based on RNNs.

### 2.5.1 multilayer Neural Networks

A more completed architecture in Artificial Neural Networks model can be the case where we have one or more hidden layers, this approach can solve much more complex learning problems in machine learning. This model consists of several layers the first layer called input layer and they receive the input data the last which is called output layer and produce the network output and the middle layers which call hidden layers because their values is not discovered in the training data set.

In the following example you can see a 3-layer neural network, in every layer there is also a bias unit which is related to an intercept term of each layer Note that bias units don't have inputs or connections going into them, since they always output the value +1:

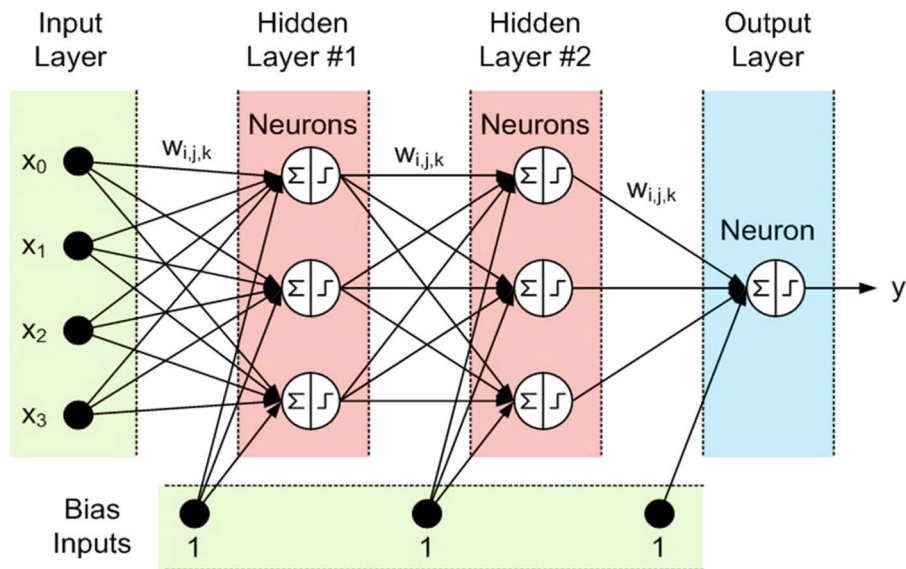


Figure 12 3-layer neural network

We have number of layers as thus  $n_1 = 3$  and every layer have its corresponding index so layer  $L_1$  is the input layer, and layer  $L_{nl}$  the output layer. Neural network is a parametric model, mainly we have 2 set of parameters weights and bias in every layer this can be shown as  $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$ , where  $W_{ij}^{(1)}$  denote the parameter (or weight) corresponding to the links between unit  $j$  in layer  $l$ , and unit  $i$  in layer  $l+1$ . And also  $b_i^{(1)}$  as the bias associated with unit  $i$  in layer  $l+1$ . Thus, in our example, we have  $W^{(1)} \in R^{3 \times 3}$ , and  $W^{(2)} \in R^{1 \times 3}$ . We also let  $s_l$  denote the number of nodes in layer  $l$  (not counting the bias unit).

For every unit in each layer we have a mean output value which is called as activation value and can be denoted as  $a_i^{(l)}$  which meant the output of unit  $i$  in layer  $l$  in the first layer the activation values for all units is equal to their corresponding inputs  $a_i^{(1)} = x_i$ . In other layers the output values can be computed based on the activation functions according to the inputs and parameters (weights and bias)  $h_W, b(x)$  can be defined as the hypothesis of this neural network that calculated as the following:

Equation 7 neural network hypothesis

$$\begin{aligned}
 a_1^{(2)} &= f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}) \\
 a_2^{(2)} &= f(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)}) \\
 a_3^{(2)} &= f(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)}) \\
 h_{W,b(x)} &= a_1^{(3)} = f(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)})
 \end{aligned}$$

A compact representation of the above formula can be representing by defining as

<sup>7</sup> [www.cs.stanford.edu/courses/soco/projects/neural-networks/Architecture/feedforward.html](http://www.cs.stanford.edu/courses/soco/projects/neural-networks/Architecture/feedforward.html)



the total weighted sum of inputs to the  $i$ th unit in layer  $l$ :

*Equation 8 Compact NN activation formula*

$$z_i^{(2)} = \sum_{j=1} W_{ij}^{(1)} X_j + b$$

$$a_i^{(l)} = f(z_i^{(l)})$$

this formula can be even in more compact representation, if we develop the activation function  $f(\cdot)$  to apply to vectors in an element-wise fashion (i.e.,  $f([z^1, z^2, z^3]) = [f(z^1), f(z^2), f(z^3)]$ ) and if we consider  $a^{(1)} = x$  to denote the values from the input layer then we can write the mentioned equations as:

$$z^{(2)} = W^{(1)}x + b^{(1)}$$

$$a^{(2)} = f(z^{(2)})$$

$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$

$$h_{W,b(x)} = a^{(3)} = f(z^{(3)})$$

Then we can write a recursive formula for activation so that given the  $(l)$ 's activation we can calculate the  $(l+1)$ 's:

*Equation 9 recursive activation NN formula*

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$

$$a^{(l+1)} = f(z^{(l+1)})$$

This is called forward propagation as one type of architecture in neural networks by changing the patterns of connections among neurons. Also by representing parameters in matrix form one can benefit from high speed linear algebra operations on computations in the networks.

### 2.6.1 Machine learning on stream data

In today's world with the growth of world wide web and social networks data generation and updating would become in the form of high speed stream in sequential order. Examples of data streams include web searches, phone conversations, ATM transactions and sensor data (31). Not enough to only perform historical analysis and batch learning techniques which is trying to learn from the entire training data at once to make a better prediction. But in cases where we have on-line stream of data we can use this sequential form of data to update prediction functions and features of data for incoming new data this can be called on-line learning when we have a stream of incoming training data which makes it impossible to learn from the whole training data set because it is updating every time. For cases like when we need to dynamically adjust to the new patterns of the data or when

data generation is a function of time, we can not only rely on the historical statistical information for prediction task but also we need to build a model with a predictive capability in real time. This requires also the data (or data features) to be informative and updated in order to be actionable immediately.

So extracting knowledge structure from these fast incoming data records can be also important, this field of science is called Data Stream Mining, here data stream as an arranged series of data samples would be processed only once or very few times by using limited computing and limited storage (26). Data stream mining can be considered a subfield of data mining, machine learning, and knowledge discovery. In this type of problem, we usually do not consider very old information or features of the data but we need up to dated features which can be helpful to discover predictive behavior of the data in the present time. Actually the aim is to predict the category labels (classes) or new values given the input data sample in the data stream (regression)

ML methods applied on stream on-line data to learn this prediction task from labeled examples by taking into account the recent history in an automatic procedure. For instance, in weather forecasting (31). Given the weather information from last few days we can estimate the weather in the tomorrow's weather, like if it has been sunny and 80 degrees the last two days, it is unlikely that it will be 20 and snowing the next day. Another property of a model in this case of ML problem is that this model should be update. Which means the model should be open to th data stream changes and updates and in other words it should get evolved with the trend of data change with time. A good example might be a retail sales model that remains accurate as the business gets larger which means in this case the subjected business is developing so our estimation on quantity of sales will not be the same as previous years.

Obviously prediction in such cases is not irrelevant to the past historical data to the process of using past training data for predicting values of unseen data is still partially true, rather to increase the performance and accuracy of the prediction we need to feed the model with last updated information on data (time series). The other important point is that the model may need to be retrained or to be updated, one solution can be exploiting incremental algorithms which learn gradually and regularly over data so that each time new training sample appears the model will update. incremental learning can be necessary when data privacy demands that instances be discarded immediately after they are seen. As an example there are incremental version of Bayesian Networks or SVM (22). Another option is to retraining the batch algorithm periodically, which seems like more optimized and straight solution because There's no buffering and no explicit retraining of the model. Periodic retraining requires more decisions and more complex implementation. However, all power of any supervised classification algorithm, in this model retraining is done only on relevant data and only when necessary.

# Chapter 3

## 3. Problem statement

In this part of our work we present both general and specific or technical aspects of the process focusing on predicting second hash-tags of tweet collected in a period of time from Expo 2015. Start with huge number of tweets from the scratch and finding and selecting the features based on predictive patterns and data behavior enhance the performance of the final prediction, although it is not enough and getting deep in semantical and syntactical parts of the language is also needed to increase the accuracy. Also applying different proper time windows and updating the feature space with regards to the upcoming events results in more precise prediction of the hashtags.

### 3.1 Problem

#### **Time-precise Prediction in Stream Micro-bloggers**

Huge quantity of information is reported and transmitted every second or millisecond in different social networks, extracting latent knowledge of this huge volume of data attract many data scientists and machine intelligence expertise to exploit these data for different analytical tasks. However recently Micro-blogging as new form of transmission where a user can report a news or status in short posts distributed by instant messages, mobile phones, email or the Web. Many people use

micro-blogging to talk about their daily events or to briefly discuss about their social or political view point or even share information.

Twitter, a well-known social network has seen a lot of growth since it launched in October 2006, provide the possibility of micro-blogging about wide range of topics and news every moment in a limited number of characters has a universal user. On average we have 6000 tweets reported on twitter every second<sup>8</sup> it shows that there is massive volume of raw information which can be exploited with Machine Intelligence sciences and extract further knowledge out of that. Although surprisingly in such micro-blogging services like twitter users may face some raw data, which needs to be processed more accurately. For example, hash-tag auto-complete in twitter will propose you several options as the suggested hash-tags, although we have noticed that these suggestions are not always so relative in this point of view that it will suggest you the top-n most popular hashtags based on recent reported tweets but it is not really always the case and these hashtags need to be updated in shorter time and in much more precise way. So by pointing to this problem, we can step into the general subjected area of this thesis which is about predicting time-precise hash-tags in short documents such as tweets with on-line stream data.

The first problem is that it can be seen as Short Text Classification not a general text categorization where we have huge number of features (words) in every document, in short text such as micro-blogs there is a strong fact that it is not easy to find the similarities between the documents because they are short and therefore not sharing enough information to clarify these similarities (Phan et al., 2008). Beside that usually there is limited number of characters per tweet so we will not have a wide choices of terms or features in every document, and this can be a great challenge to short text classification (STC) because the more efficient features in every input document (tweet) the more accurate the final output of the ML algorithm.

---

8 <http://www.internetlivestats.com/twitter-statistics/>

## 3.2 Case study

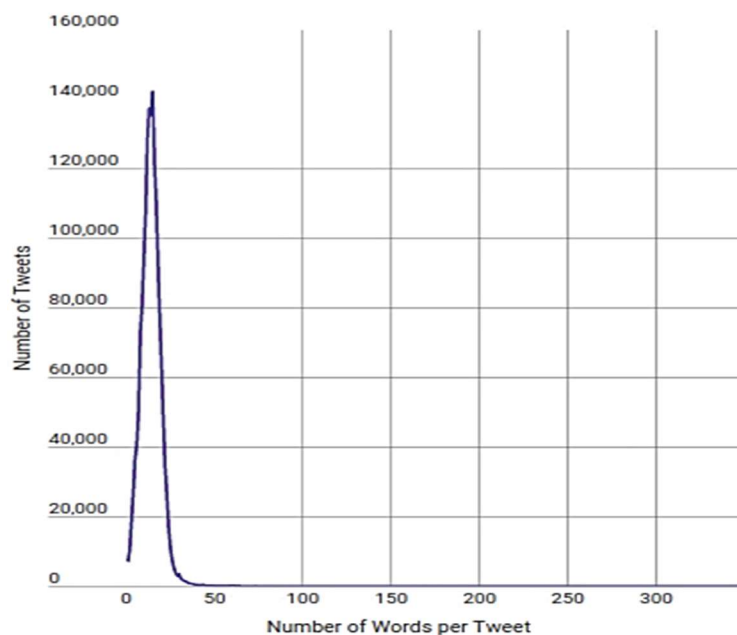
### Power of time window in hashtag prediction in twitter

Generally, in this thesis the work is on a specific collection of tweets from “Expo 2015 Milan” collected in 4 months, the following table present a brief information on the dataset:

*Table 1 Expo2015 Dataset Overview*

|    | Entities                        | Quantity |
|----|---------------------------------|----------|
| 1. | Number of Users                 | 811615   |
| 2. | Number of Tweets                | 2191905  |
| 3. | Number of all Hashtags/Entities | 364999   |
| 4. | Number of Hashtags              | 267540   |
| 5. | Max(tweets) Per Day (day 91)    | 27331    |
| 6. | Max(tweets) for a User          | 75543    |
| 7. | Number of Valid Tweets          | 1732978  |

The goal is to predict the second hashtags as the output value based on the given input data which are words (features) before occurrence of the second hashtag. In the following you can the trend of words numbers in each tweet, which represents that in most of the cases there is less than 40 words per tweet in our dataset:



*Figure 13 Frequency of number of terms in tweets*

It can clearly reflect the problem of finding informative data pattern and proper features out of these limited number of word options in every tweet, so that they can act sufficiently in predicting the second hashtags. However, this textual format of words in each tweet is not the proper input format of the data for Machine learning algorithms. As the foundation of ML is based on mathematical and statistical techniques we need to feed these algorithms with numerical representation of the input data, a proper textual representation can be to replace each word with its frequency in the whole document and then transmit the whole document in the vectorial feature space.

Now with this proper data representation of the document we can define the problem in machine learning point of view. The main task can be predicting the second hashtag of tweets so that all words appear before the second hashtag plus the first hashtag has been considered as features in the test set. There were 2 main issues we faced, the first one was to define a ML method coping with upcoming new features which can update these features and parameters properly, the other problem was to learn syntactical relationship among features in document and understand the time-varying predictive behavior of the second hashtags. By focusing on these two problems and finding appropriate solutions (will explain in detail in the following sections) we reach a better and more accurate result with almost 40% progress in the final prediction comparing other approaches already have been tested on the same dataset.

There has been already another research done on the same dataset with two methods the first one is the TOP-TAG which predict the second hashtag just based on the popularity or occurrence of the second hashtags this is almost what twitter is using to suggest hashtags, although it has a very low precision, and doesn't reflect accurate result among all other methods has been tried on this set of tweets. The other method has been tested is co-occurrence, feed by the first hashtag as its feature and is supposed to predict the second hashtag of tweets. Although this method has been used widely for several years in many text processing tasks and also in this case it represents much better result with regarding to TOP-TAG approach, but it is not considering any semantical or syntactical aspect of the words in documents (tweets). Based on the final results in this thesis, it has been demonstrated the process we followed has overcome the previous works on this dataset, the key points in this achievement can be briefly explained as getting deeper in semantical layers of the features using embedding techniques and properly update the stream data to be flexible in predicting second hashtags based on new upcoming events which can also have effect on the semantical relation of the words. It clearly displays that there need to be a more updated approach which can consider trend of the data stream and also be sensitive to the dynamic semantical relation among similar words.

However it was not enough just to apply the ML algorithms on the prepared features vectors to predict only one word as the second hashtag the output error rate has no progress with respect to the previous approach (co-occurrence), although by

exploiting embedding techniques we touch the higher level semantic of data means that we group-wise the output values with regards to the whole document words so that rather than learning and predicting only one hashtag as the output our approach is learning the similar words of the predicted value so that the output value is not a single value but a vector of 10 similar words.

On the other hand, one important fact about this stream data can be its dynamic time-varying behavior. Let's clarify this with an example, imagine that in second month of Expo2015 there was a famous wine festival in Italian pavilion only for one week, so before date a feature like "#Wine\_Festival" might not exist or it might have very low frequency and efficiency among other features, so after the first periodic update on the feature space this feature (plus any other new term or hashtag) will be added (if not existed) to the feature spaces and the syntactical relation among features will be updated or rebuilt based on the corresponding time window, this will keep on top frequent hashtags and update their similarities based on the embedding techniques.

Another fact that we should consider was the tweets from different languages, this will make problem when we are in preprocessing step or more precisely when we are involved with morphological aspects of the text which is language dependent in some cases, for example we need to extract the stem of the terms there is different modules for different languages that do such task in Python for different languages but we need to clarify the language to use a proper approach. However, in many cases tweets are multi-language where hashtags or words are from more than one language. In this thesis we investigate 2 cases the first one when only English tweets are considered and the other one when we identify both English and Italian tweets and separately apply preprocessing methods on them but they were all in a unit training set so that "#milano" and "#milan" were considered as similar words, although the first case represent higher precision in the final result.

# Chapter 4

## 4. Problem Solving

So far we try to have a brief overview of the thesis process and several problems we faced with the nature of the data but to get more in detail and to get involved with more technical and practical aspects of the process we can start by a higher level aspect of the problem and the corresponding solution.

### 4.1 System Architecture

System architecture of the process can be seen as follows where the incoming tweets first will get clean and converted into proper feature representation for machine learning algorithms, during this steps different libraries, tools and techniques have been used and the most accurate ones with regard to the results and the data behavior have been used.

Preprocessing can be explained as a customized process in this thesis based on the nature of the data and different outlier's emoji's, needless URLs... etc. each of these cases were identified and removed or in some cases replaced with a standard value. Although the most important issue of this dataset was to choose a policy about tweets in different languages or even multi language tweets where we have tweets with words or hashtags from more than one language this makes problem in stages like applying stop words or extracting stemming of the words where we need to specify the languages more specifically. Different methods like bag-of-words or different libraries and modules of Python like NLTK, detectlanguage, Gensim have been used.

In the embedding part based on the considered window length we update the feature space and corresponding similarities and semantical relationships among



words and reproduce the corresponding vector for each feature term. Technically we have compared 2 different modules in Python for this purpose, Word2vec and Glove. Word2vec is released on 2013 by a team of researchers led by Tomas Mikolov at Google, this module has been implemented in Python by Gensim modeling tool. Glove is another embedding method which is more based on statistical methods rather than neural networks, released in 2014 by Pennington et al. from Stanford. In both methods the goal is to draw semantic and syntactic patterns among words which can be reproduced using vector arithmetics and in this thesis we are mainly training them based on the whole corpus to find the relationship among words and then group-wise or cluster the output of the problem so that the output of the problem is a vector of similar hashtags rather than only one single hashtag meaning that rather than learning to predict the second hashtag the model is learning to predict also similar words of the predicted value.

In the section related to ML we have 3 sets of data training set, validation set and testing set which by using Cross-Validation of scikit-learn in Python we randomly split data for these sets to tune parameters (with validation set) and prevent over-fitting by holding out portion of available data as test set for evaluating the model with unseen data. Also in machine learning section we apply time windows with different lengths which were equal or greater than the time window applied on the embedding part. Finally by comparing different methods such as Support Vector Machine (SVM), Naive Bayesian (Gaussian NB), K-Nearest Neighbor (KNN) and Linear Discriminant Analysis (LDA)

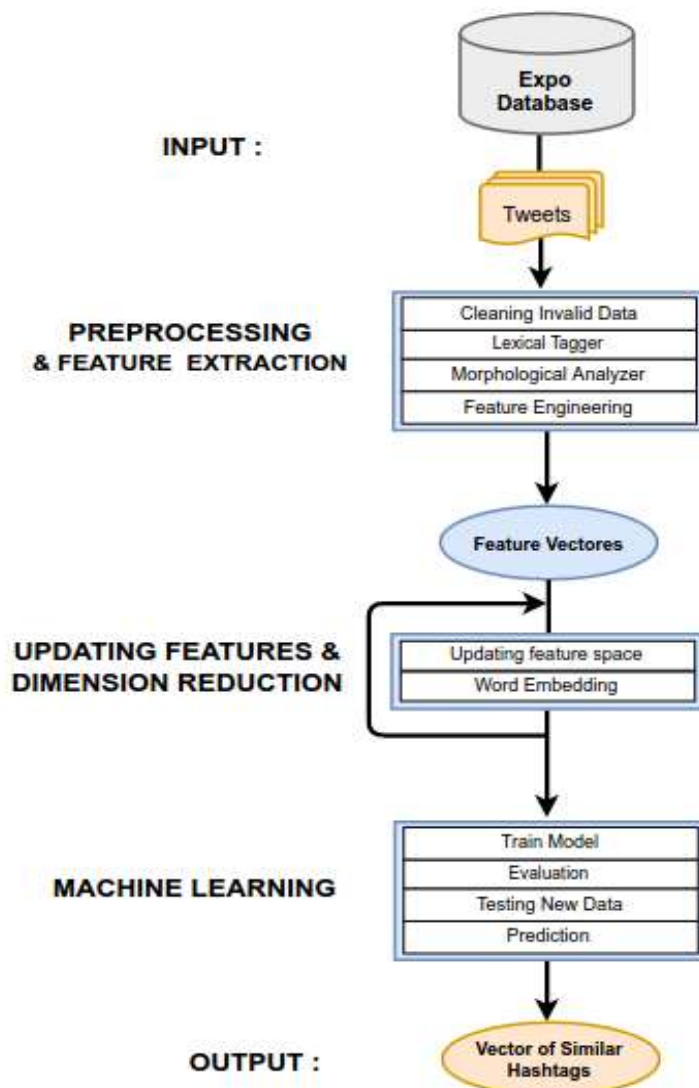


Figure 14 System architecture overview

## 4.2 Dataset

the data we have used in this thesis is a collection set from Expo2015 there are 4 main tables with information about authors, tweet contexts and time-stamps, another table related to hashtags occurred in each tweet and finally the one which is related to the hashtags and a Boolean attribute representing whether they are also semantic web entities or not. In macro-events cases like this, tweets are consisting of self-generated contents reporting many different small events centered around specific themes, festivals, time or places. In order to apply several machine learning tasks, it is important to have enough statistical knowledge about the data set which can give a better view to explain the further predictive behavior of the data. In the following several statistical information has been represented about the Expo2015 dataset, expo started from 1<sup>st</sup> of May for almost 6 months but the data has been collected from 19<sup>th</sup> of June up to 30<sup>th</sup> of October.

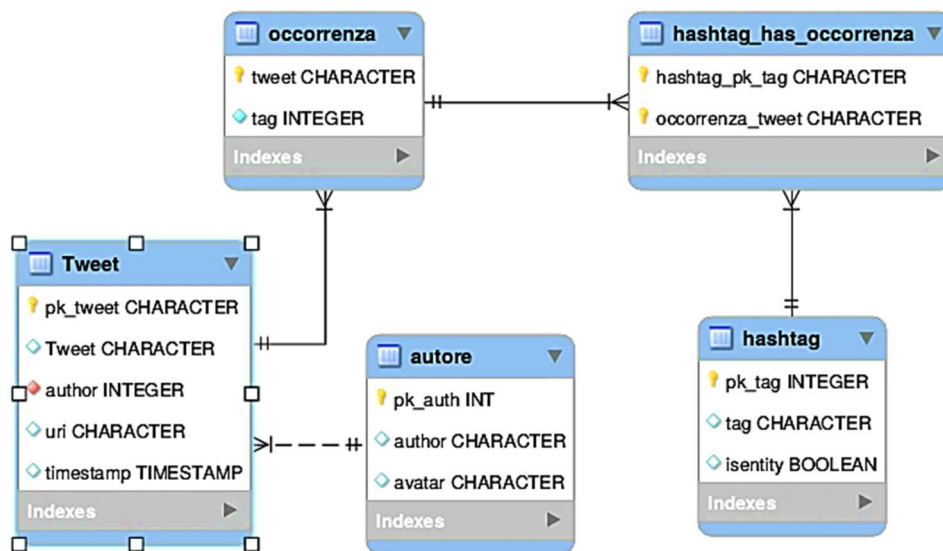


Figure 15 Expo2015 Data Tables

From the following diagram you can see the frequency of the tweets reported from day 53<sup>rd</sup> to 182<sup>nd</sup> which represent special trend in different days, rather than day 8 different days that the system has been down for several hours or almost for the whole day in day 123<sup>rd</sup> but in general the figure explain that the number of tweets per day is not always the same and doesn't have a steady trend all the time but also it fluctuated a lot which might have different reasons like number of visitors or holidays but it might represent that in special days because of important events number of tweets will increase reporting this event in a specific time-stamp, and this diagram shows that there is an upward trend in the number of tweets reported every day.

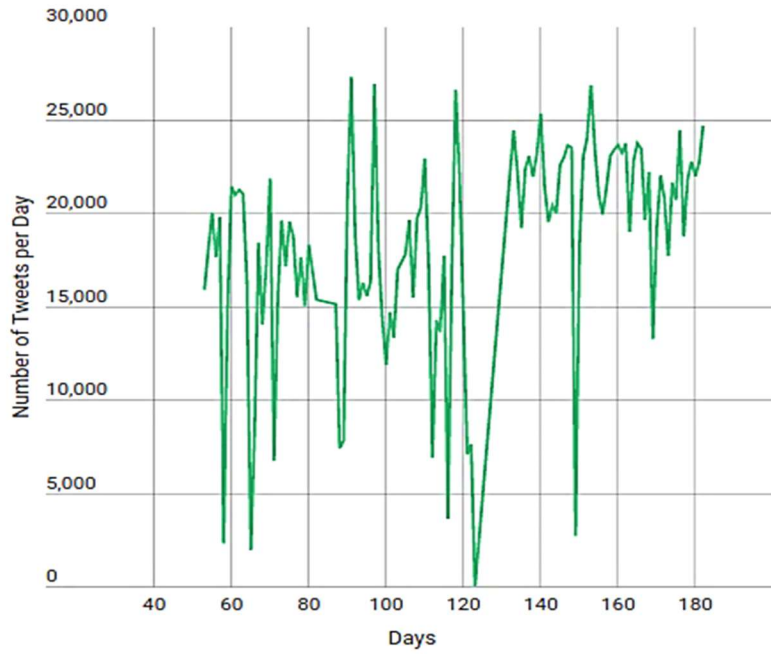


Figure 16 Frequency of tweets

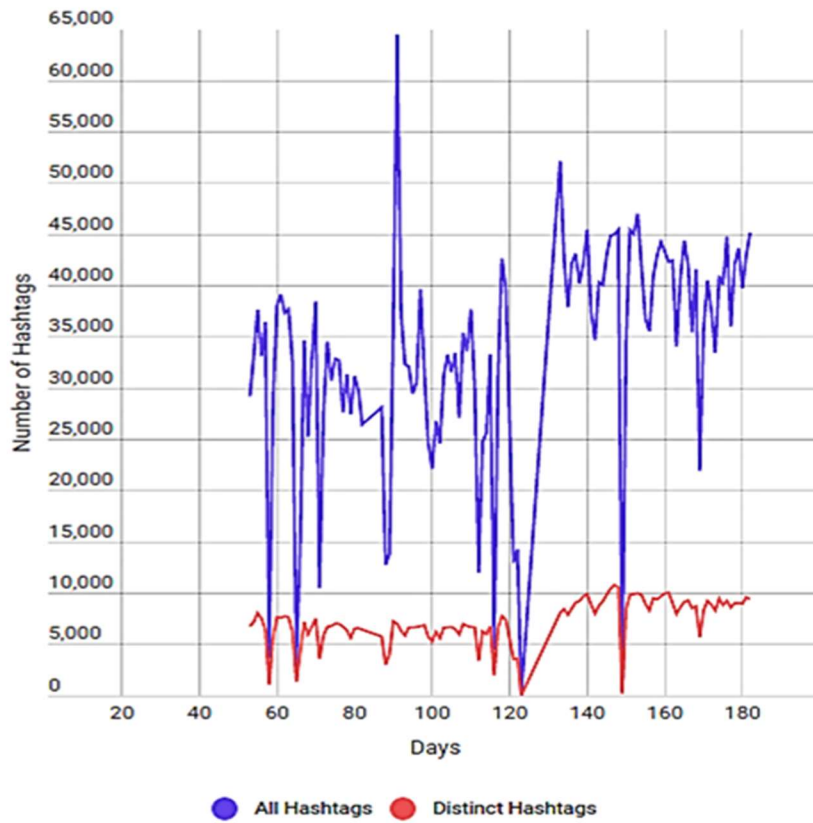


Figure 17 hashtags daily frequency

Another important fact about the data which can be important statistical information with regards to the thesis problem is the most popular hashtags used in the whole tweets, this can be important to evaluate or interpret the final prediction results also the embedding vectors. In the top 20 relevant hashtags similar words in different languages exists like “#italy” and “#italia” from these we can claim that such words should be appear in the same similarity vectors if we are supposed to consider Italian, English and multi-language tweets. Although in this thesis after identifying tweets languages only Italian and English tweets where selected to be examined and other tweets where eliminated.

*Table 2 Most Frequent hashtags in Expo2015*

|     | Quantity | Hashtags           |
|-----|----------|--------------------|
| 1.  | 483938   | "#expo2015"        |
| 2.  | 192940   | "#expo"            |
| 3.  | 79707    | "#milano"          |
| 4.  | 61965    | "#expomilano2015"  |
| 5.  | 33881    | "#milan"           |
| 6.  | 22294    | "#news"            |
| 7.  | 19394    | "#italy"           |
| 8.  | 18841    | "#expomilano"      |
| 9.  | 17066    | "#food"            |
| 10. | 14665    | "#인포니트"            |
| 11. | 11651    | "#art"             |
| 12. | 11624    | "#periscope"       |
| 13. | 11383    | "#italia"          |
| 14. | 9560     | "#nationalday"     |
| 15. | 8994     | "#mtvema"          |
| 16. | 8860     | "#alberodellavita" |
| 17. | 8437     | "#nominothekolors" |
| 18. | 8388     | "#foodporn"        |
| 19. | 8071     | "#vixx"            |
| 20. | 7534     | "#paris"           |

The following diagram which represent the distribution of tweets per user is also representing an important fact that almost all of the users who tweeted in this dataset were ordinary visitors who has reported the events of that day in Expo so the trend of the predictive behavior in the data is not affected by these users' tendency but mostly by events and attractions focused on the Expo macro-event.

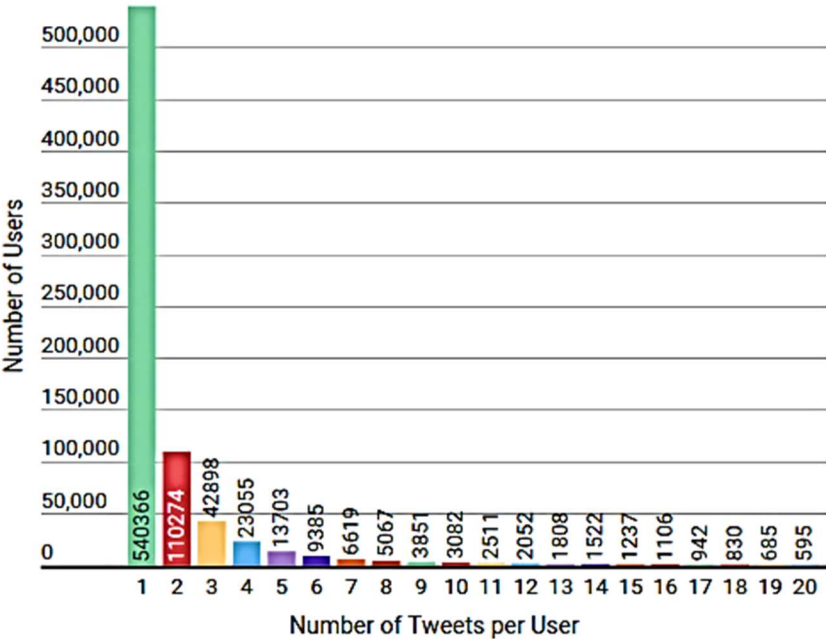


Figure 18 distribution of tweets for each user

Last diagram of the statistical information shows number of hashtags in every tweet which shows that the majority of tweets have only one hashtag although we are supposed to predict the second hashtags so we also eliminate all tweets containing only one hashtag which are to the number of 784283, but it was not the only restriction applied on the tweets, also length of tweets was checked to be in the standard size of 140 characters.

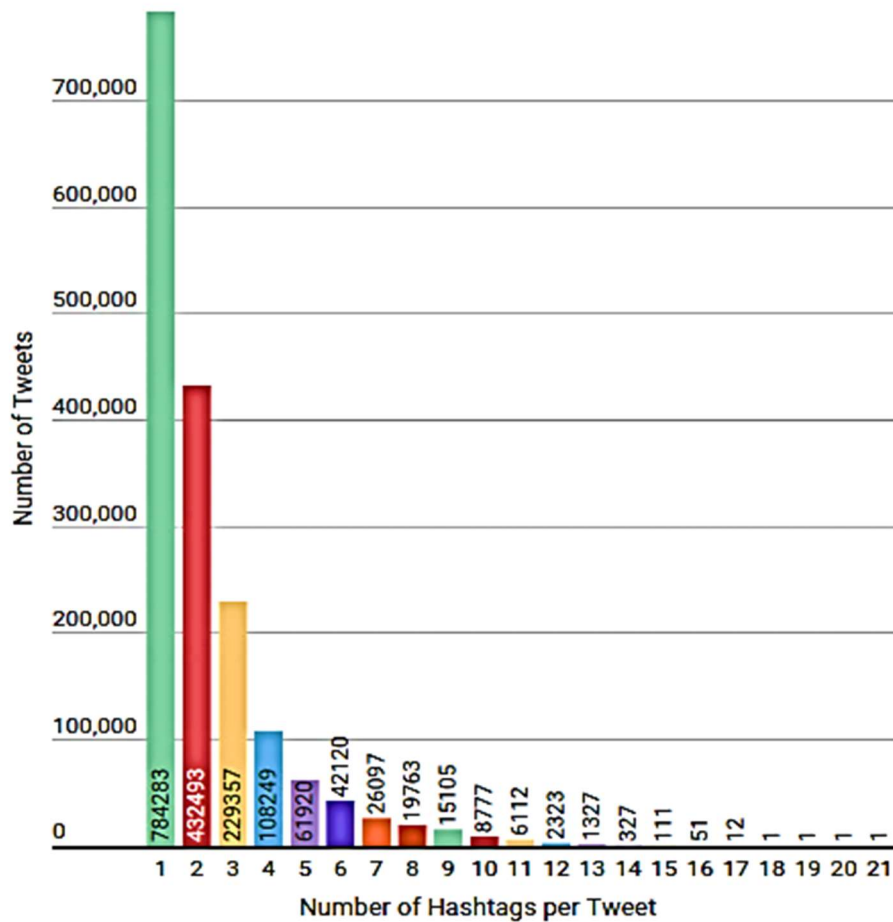


Figure 19 hash-tag distribution in tweets

## 4.3 Cleaning Data & Text Preprocessing

This process has several steps starting with cleaning data from redundant terms, outliers or uninformative words in each tweet, this process is also known as preprocessing of the data which needs to be done carefully because there are few words in every tweet and it is important, not to eliminate the useful words, remaining words will be used as features of machine learning algorithms. However, it is not possible to use directly these terms as features. In order to use a textual data as input data to a ML algorithm we need to convert it into a standard data representation format for ML. We need to make a numerical representation out of each term (feature) and project it in a huge scalar feature space so that each term in a document (tweets) is represented as a sparse vector of features and the corresponding numerical value of that feature is in fact the frequency of this word in the document. Not enough yet we still have to work in the extraction and selection of these features to make sure that they can act properly in enhancing the final performance in ML. By cleaning data, we mean to deal with numbers, punctuations and stop-words for these issues NLTK and different Regular Expressions have been used to solve these problems, for many reasons and Unicode problems, it makes sense to remove punctuations and emoji's. On the other hand, in this case, we might need to cope with a sentiment analysis problem, and it is possible that "!!?" or ":-\" could carry sentiment, and should be treated as words. In this thesis, for simplicity, we remove the punctuation altogether, but it can be an interesting topic for future work in this regards. Also about numbers in this thesis they are treated as words. There were also other strings which were not in ASCII format and using different regular expressions they were replaced. To eliminate these emoji's and useless strings, we will use a package in python for dealing with regular expressions, called `re`.

### 4.3.1 Stop Words

After splitting and tokenization of words in every tweet we need to remove stop-words. Stop-words are defined as words which occur frequently. In many cases stop-words are critical because of the fact that, by eliminating words which are often used in a language rather we can focus on the important words. Several groups of stop-words exist which can have different level of importance and meaning in different applications. For example, in some applications removing all stop words right from determiners (e.g. the, a, an) to prepositions (e.g. above, across, before) to some adjectives (e.g. good, nice) can be an appropriate stop word list. A minimal stop list consisting of just determiners or determiners with prepositions or just coordinating conjunctions depending on the needs of the application. Example of minimal stop-word lists can be the following:



- **Determiners**- Determiners tend to mark nouns where a determiner usually will be followed by a noun  
examples: the, a, an, another
- **Coordinating conjunctions**– Coordinating conjunctions connect words, phrases, and clauses  
examples: for, and, nor, but, or, yet, so
- **Prepositions**- Prepositions express temporal or spatial relations  
examples: in, under, towards, before

Also in general there are lists of stop-words already have been published

**Snowball stop-word list:** this stop word list is published with the Snowball Stemmer

**Terrier stop-word list:** this is a pretty comprehensive stop word list published with the Terrier package.

**Minimal stop word list:** this is a stop word list that I compiled consisting of determiners, coordinating conjunctions and prepositions

**Construct your own stop word list:** this article basically outlines an automatic method for constructing a stop word list for your specific data set (e.g. tweets, clinical texts, etc.)

#### *4.3.2 Bag-of-words*

By exploiting Information Retrieval (IR) and Natural Language Processing it is possible to model the text (or image) into a multi-set (bag) of words. This model which widely has been used in text categorization and computer vision is called bag-of-words (32). It is a standard text representation that consists of representing each document (in this thesis by document we mean every tweets) by words occurred in that document in fact frequencies of occurrence of each word is used as the features of the data samples (33). The steps of this method can be briefly explained as the following.

### ***Remove non-letters***

- *Cleaning data by using regular expressions to do a find-and-replace*

### ***Convert to lower case, split into individual words, apply stop-words***

- *Convert to lower case*
- *Split into words*
- *Download text data sets(NLTK), including stop words*
- *Import the stop word list*
- *Remove stop words from "words"*
- *Join the words back into one string separated by space*

### ***Initialize the "CountVectorizer" object, which is scikit-learn's bag-of-words tool***

- *Use fit\_transform() feeding with a list of tweets as input values*
- *fits the model and learns the vocabulary*
- *transforms our training data into feature vectors*
- *convert the result to an numpy array*

### ***4.3.3 Transforming Text Documents into Vector Space***

Text documents in their original form are not possible to learn from. They must be transformed to match the learning algorithm's input format which is an attribute-value representation; this means that we need to transform the documents into a numerical vector space.

After pre-processing of the tweets which has already explained, containing stop-word filtering for omitting frequent and meaningless words (e.g. a, an, this, that), word stemming for reducing the number of distinct words, lowercase conversion etc. after that transformation takes place. Each word or more precisely each feature will correspond to one dimension, identical words to the same dimension. Let the word  $w_i$  correspond to the  $i$ th dimension of the vector space. The most commonly used method is the so-called TF•IDF term-weighting method. Denote  $TFIDF_{(i,j)}$  the  $i$ th coordinate of the  $j$ th transformed document.

*Equation 10 TFIDF/IDF formula*

$$TFIDF_{(i,j)} = TF_{(i,j)} \cdot IDF_{(i)}$$
$$IDF_{(i)} = \log \frac{N}{DF_{(i)}}$$

Where  $TF_{(i,j)}$  means how many times does the  $i$ th word occur in the  $j$ th document,  $N$

is the number of documents, and  $DF_{(i)}$  counts the documents containing the  $i$ th word at least once.

The transformed documents together form the term-document matrix. It is desirable that documents of different length have the same length in the vector space, which is achieved with the so-called document normalization:

*Equation 11 normalized TFIDF*

$$TFIDF'_{(i,j)} = \frac{TFIDF_{(i,j)}}{\sqrt{\sum_i TFIDF_{(i,j)}^\beta}}$$

The dimensionality of the vector space may be very high, which is disadvantageous in machine learning (complexity problems, over-fitting), thus dimension reduction techniques are called for. In this thesis we have a vocabulary of size 10000 words, although main problem was to reduce the dimensionality of the class labels or second hashtags where word embedding takes into account. Two possibilities exist, either selecting a subset of the original features, or merging some features into new ones, that is, computing new features as a function of the old ones.

Two embedding techniques Word2vec and Glove have been used and explained more in detail in the followings, these techniques have been only applied to reduce the dimensionality of the predicted values which enhance the accuracy and precision of the final result up to 19%. comparing these two exploiting techniques they both have a almost equally a good result, although Word2vec have a higher precision (around 3%-4%) but Glove appear to be a bit faster and needless to say that Glove is a kind of open-source approach which can be attractive in order to be used for free. In the following embedding approaches explained more technically.

## 4.4 Word2Vec

This approach has been used for learning word vectors in huge dataset where we have very large number of words and massive vocabulary size. None of the previously proposed architectures has been successfully trained on more than a few hundreds of millions of words, with a modest dimensionality of the word vectors between 50 – 100. In this case not only similar words are close to each other in words space but also they can different degree of similarity, a concept which has already been used in the morphological languages, there can be several constant rules like the fact that nouns can have different ending parts and searching for similarities in subspace of the original vector space will end to words with same endings. It was found that similarity of word representations goes beyond simple syntactic regularities and surprisingly results have been achieved by using simple algebraic operations, performed on the word vectors, it was shown for example that:

vector ("King") – vector ("Man") + vector ("Woman") results in a vector that is closest to the vector representation of the word Queen.

Word2vec by developing new model architectures that preserve the linear regularities among words by measuring both syntactic and semantic regularities, this method showed that many such regularities can be learned with high accuracy which is depends on the dimensionality size of the word-vectors and the amount of training data.

### 4.4.1 Word2vec Architecture

Different types of models including the famous Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA) were proposed for computing continuous representations of terms in documents. word2vec focus on distributed representations of terms in document learned by neural networks, it has been already proved that other techniques perform significantly better than LSA for preserving linear regularities among words; LDA moreover becomes computationally very expensive on large data sets (34).

In the following the comparison between different architectures and to this purpose with different model architectures first step is to define the complexity of a model as the number of parameters that need to be accessed to fully train the model. Next step is maximizing the accuracy, and at meanwhile minimizing the computational complexity. For all the models we are investigating, the training complexity is proportional to

$$O = E \times T \times Q,$$

Where E is number of the training epochs, T is the number of the words in the training set and Q is defined further for each model architecture. Common choice is E = 3 – 50 and T up to one billion. All models are trained using stochastic gradient descent and back propagation.

### **Feed-forward Neural Net Language Model (NNLM)**

The probabilistic feed-forward neural network language model consists of input, projection, hidden and output layers. The process is in this way that at the input layer,  $N$  previous words are encoded applying one of  $V$  coding, where  $V$  is size of the vocabulary. The input layer is then projected to a projection layer  $P$  with the dimensionality of  $N \times D$ , using a shared projection matrix. As only  $N$  inputs are always active, composition of the projection layer is a cheap operation.

Complexity of NNLM can be high for computation between the projection and the hidden layer, as values in the projection layer are dense. Consider a typical example with usual amount of  $N = 10$ , the size of the projection layer ( $P$ ) might be 500 to 2000, and the hidden layer size  $H$  is usually 500 to 1000. Moreover, the hidden layer is used to compute probability distribution over all the words in the vocabulary, resulting in an output layer with dimensionality  $V$ . Thus, the computational complexity per each training example is

$$Q = N \times D + N \times D \times H + H \times V,$$

Most of the complexity is caused by the term  $N \times D \times H$ . In these models, hierarchical softmax have been used where the vocabulary is represented as a Huffman binary tree. This is based on the previous studies where class estimation is according to the words frequencies in neural net language models. Huffman trees dedicate small binary codes to frequent words, and it later reduces the number of output units that need to be evaluated: in comparison with Huffman tree method balanced binary tree needs  $\log_2(V)$  outputs for evaluation part but the Huffman tree based requires only  $\log_2(U \text{ nigram perplexity } (V))$ . imagine a huge size of vocabulary like one million words this method reduce the speed like two times faster but the bottleneck of time consumption is in the  $N \times D \times H$  however in the following architectures by reducing the hidden layer this problem will be solved and efficiency of softmax normalization will be bold.

### **Recurrent Neural Net Language Model (RNNLM)**

To cover come limitations in the feed-forward Neural Network Language Model another architecture model has been proposed, constrains such defining the context length (the order of the model  $N$ ), and because of the fact that in theory RNNs can efficiently represent more complex patterns than the neural networks with few number of hidden or projection layers. The RNN model has no projection layer; only input, hidden and output layer. Special characteristic point in this model is the recurrent matrix that connects hidden layer to itself, exploiting time-delayed links, which let this model to form some kind of short term memory, as information from the past can be represented by the hidden layer state that gets updated based on the present input and the state of the hidden layer in the former time step (35). The complexity of the RNN model is

$$Q = H \times H + H \times V,$$

Where the word representations  $D$  have the same dimensionality as the hidden layer  $H$ . Also in this case, the term  $H \times V$  can be reduced to  $H \times \log_2(V)$  by using hierarchical softmax. What is clear again is that most of the complexity comes from  $H \times H$ .

#### **Parallel Training of Neural Networks**

for the training phase of word2vec several models on top of a distributed framework called DistBelief, including the feed-forward neural networks have been used. Multiple copy of the same model can be executing in parallel with this framework and each section synchronizes its gradient updates through a centralized server that saves all the parameters (35). In fact, training phase is done in parallel processes (can be hundreds or more) and mini-batch asynchronous gradient descent with an adaptive learning rate procedure with the name of Adagrad has been exploit.

#### **New Log-linear Models**

in general, 2 architectures are used in wor2vec which has been inspired from the already explained architectures used in the learning process of distributed representations of words trying to reduce the complexity. As already explained non-linear hidden layers in the model are caused the majority of the complexity problem in the model but these layers are the main reason to keep Neural Networks model so active so the effort in word2vec is to simplifies the neural network representation but enhance the overall performance of the training process.

Following the already explained architectures and the fact that neural network language model can be successfully trained in two steps: first, continuous word vectors are learned using simple model, and then the N-gram NNLM is trained on top of these distributed representations of words. Two methods are the followings:

##### **4.4.1.1 Continuous Bag-of-Words Model**

The first proposed architecture is working almost the same as the feed-forward neural networks, where the non-linear hidden layer is removed and not only the projection matrix but also the projection layer is shared for all words; thus, all words get projected into the same position (their vectors are averaged). This architecture is called a bag-of-words model as the order of words in the history does not affect by the projection. However, this method can be updated so that we also use words from the future. Training complexity is then

$$Q = N \times D + D \times \log_2(V)$$

this model has also been denoted as CBOW, as unlike standard bag-of-words model, it uses continuous distributed representation of the context. The model architecture is shown at Figure 20<sup>9</sup>. Note that the weight matrix between the input and the

---

<sup>9</sup> [www.deeplearning4j.org/word2vec](http://www.deeplearning4j.org/word2vec)

projection layer is shared for all word positions in the same way as in the NNLM.

#### 4.4.1.2 Continuous Skip-gram Model

This architecture is working in a way that the goal is to maximize the classification of a word based on the other terms in the same sentence and unlike CBOW it is not supposed to estimate the current term based on the document or context. So that the current term is considered as the input value and it will be feed to a linear classifier with continues projection layer and predict words within a certain range before and after the input term. We found that enhancing the range improves performance of the resulting word vectors, although beside these benefits computation complexity is also increased. Since words which are relatively in far distance with regards to the input term are usually not so related to this word than those who are closer to it and much more relevant, we give less weight to the distant words by sampling less from those words in our training examples. The training complexity of this architecture is proportional to

$$Q = C \times (D + D \times \log_2 (V))$$

where  $C$  is the maximum distance of the words. Thus, if we choose  $C = 5$ , for each training word we will select randomly a number  $R$  in range  $< 1; C >$ , and then use  $R$  words from history and  $R$  words from the future of the current word as correct labels. This will require us to do  $R \times 2$  word classifications, with the current word as input, and each of the  $R + R$  words as output.

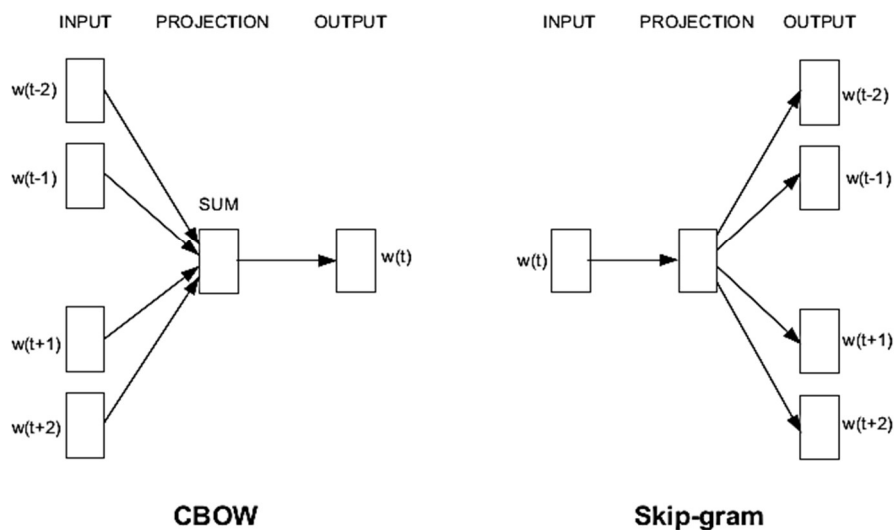


Figure 20 word2vec architecture overview

#### 4.4.2 GloVe (Global Vectors for Word Representation)

Another embedding technique has been used in this thesis is GloVe an open source academic tool from Stanford which has been acting almost well in embedding tasks. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. The training phase is exploit on aggregated global word to word co-occurrence statistics from a corpus (36), the output of the problem is constructed from a linear substructure of the word vector space. GloVe algorithm consists of following steps (37):

1. Collect word co-occurrence statistics in a form of word co-occurrence matrix  $X$ . Each element  $X_{ij}$  of such matrix represents measure of how often word  $i$  appear in context of word  $j$ . usually we scan our corpus in the following manner: for each term we look for context terms within some area, a `window_size` before and a `window_size` after. Also we give less weight for more distant words, usually using the formula  $decay=1/offset$

2. Define soft constraints for each word pair:

$$w_i^T w_j + b_i + b_j = \log(X_{ij})$$

Here  $w_i$  - vector for the main word,  $w_j$  - vector for the context word,  $b_i, b_j$  are scalar biases for the main and context words.

3. Define a cost function

$$J = \sum_{i=1}^V \sum_{j=1}^V f(X_{ij}) (w_i^T w_j + b_i + b_j - \log X_{ij})^2$$

Here  $f$  is a weighting function which helps us to prevent learning only from extremely common word pairs. The GloVe authors choose the following function:

$$f(X_{ij}) = \begin{cases} (X_{ij}/X_{max})^\alpha & \text{if } X_{ij} < X_{max} \\ 1 & \text{otherwise} \end{cases}$$



# chapter5

## 5. Implementation Experience

In this chapter we will get more in detail with the practical aspect of the work and represent the result of work step by step to follow the process technically. This thesis has been done with Python Language Programming, due to its popularity as a general purpose programming language, as well as its adoption in both scientific computing and machine learning. Different specific Packages in Python which are powerful open-source libraries facilitating practical machine learning. Some of the main libraries exploited in this thesis such as Numpy for handling N-dimensional and sparse matrices, Pandas for data analysis tasks, including structures such as dataframes, Scikit-learn with the machine learning algorithms used for data analysis and data mining tasks and Matplotlib for 2D plotting, producing publication quality figures.

There are many other libraries and modules used for different purposes in Python like NLTK, Gensim, Langdetect and etc. Several of these libraries and their functionality in this thesis will discuss in the following.

### 5.1 Multi-language tweets preprocessing

From the starting step preprocessing we were facing important issues about tweets languages, the problem is that in text preprocessing there are different libraries in Python which can help you in different steps like tokenization, stemming and lemmatization, parsing and etc. but these functions which are related to lexical or morphological aspects of the text have several sets and language specified rules, which need to specify the language of the context first (38).

The goal of both stemming and lemmatization is to reduce grammatical and morphological forms and sometimes derivation forms of a word to a common base form. For instance:

am, is, are →be

art, arts, artistic, arts' →art

The result of this mapping of text will be something like:

we are in #expo2015 beautiful #design #arts→

we be #expo2015 beauty #design #art

Although these two functions are acting in different ways but following the same goal. *Stemming* usually refers to a raw heuristic process that chops off the ends of words to achieve this goal correctly most of the time, and often includes the removal of derivational affixes. *Lemmatization* usually refers to tasks related to vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, this form of word is called the *lemma*. If we face with the token *saw*, stemming might return just *s*, whereas lemmatization return either *see* or *saw* depending on whether the use of the token was as a verb or a noun. The two may also differ in that stemming most commonly collapses derivationally related words, whereas lemmatization commonly only collapses the different inflectional forms of a lemma. Anyway we need a deep grammatical and morphological knowledge for each language to be able to produce either lemma or stems out of the words in document, in fact in practice many approaches can perform such tasks for English or several other languages context although there are not such tools for all languages or not all of them may act perfectly good because it needs strong dictionary source and precise information about grammar rules of every specific language.

In Expo dataset with tweets reported by different languages like English, Italian, Arabian and etc. so the issue is dealing with lemmas and stems replacement of words in morphological and lexical steps of the preprocessing. So we need to set the strategy how to deal with tweets in different languages, one problem is that there is no already defined tools in python to perform the lemmatization and stemming, for example NLTK which is known as the most powerful tool for this purpose can support the following languages: Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Norwegian, Portuguese, Romanian, Russian, Spanish and Swedish. So it is obvious that in case we cannot find the proper tool in python for this case we need to define a set of rules and stem words for every language by ourselves which is beyond the scope of this thesis, beside there might be negligible number of tweets in other languages rather than English or Italian which are the main languages of tweets in this dataset. This little number of tweets cannot be enough to be included in the training set and the ML approach cannot learn properly in this situation, so error raise is irrefutable. So language setting in this thesis is about to keep the Italian and English tweets and eliminate others, although we examine to keep the French and German tweets but this cause the error increment so we only consider the major languages of this dataset.

This procedure was impossible or less accurate without tools with strong language specified datasets of vocabulary and grammar rules, so detecting the language of a given context is a must, in this thesis we used an API with the name DetectLanguage

released by MIT which provides both free and premium service. They accept to provide the best premium service for more than one month for free for this thesis. This API now can detect 160 languages.

Detectlanguage API appears helpful specifying the language of tweets and applies a proper localized morphological tool. Based on the explained strategy only Italian and English tweets were selected we examine 3 different cases, first to only accept English tweet and estimate the result based on this hypothesis, second to also accept Italian tweets but treat them in 2 different training set and the last setting was to apply proper preprocessing techniques on Italian and English tweets separately but consider them as a unit training set. Based on the process we were following the third setting seems to be more close to our goal because we could find words like “#italy” and “#italia” in the same similarity groups at the embedding step.

For example, with the following code we can get the following results:

```
puncList = [".", ";", ":", "[", "]", "!", "?", "/", " ", "@", "$", "&", ")", "(", "\'"]
try:
    lan= detectlanguage.simple_detect(features)
    print lan

    if (lan=="en"):
        for punc in puncList:
            features1=features.replace(punc, '')
            lower_case = features1.lower()
            words = lower_case.split()

            stemmer = SnowballStemmer("english") # Choose a language
            words="".join([stemmer.stem(word) for word in words])

            stops = set(stopwords.words("english"))
            words = "".join([str(w) for w in words if not w in stops])

elif(lan== "it"):
    for punc in puncList:
        features1=features.replace(punc, '')
        lower_case = features1.lower()
        words = lower_case.split()
        stemmer = SnowballStemmer("italian") # Choose a language
        words= "".join([stemmer.stem(word) for word in words])
        stops2 = set(stopwords.words('italian'))
        words = "".join([str(w) for w in words if not w in stops2])
```

This way with a tweet like:

*“Finally we agreed to visit #Japan #padiglione #Expo2015 #Milano #arts”*

→*“Final agre visit Japan pavilion padiglione Expo2015 Milano art”*

## 5.2 Data to Vector

After normalizing the context by selecting the normalized form of words (stem or lemma) which carry essential information and eliminate vain form of terms and stop-words from tweets. Then it is time to convert each tweet represented as a list of tokens (stems), into an appropriate numerical vector form suitable for machine learning algorithms. This task can be done in three essential steps, in the bag-of-words model:

1. counting number of times, a word appears in each message (term frequency)
2. weighting the counts, so that frequent tokens get lower weight (inverse document frequency)
3. normalizing the vectors to unit length, to abstract from the original text length (L2 norm)

Each vector has as many dimensions as there are unique words in the Expo2015 corpus:

```
#*****DATA VECTOR*****
tweets = pandas.read_csv('../tweetvl.csv', sep='\t', quoting=csv.QUOTE_NONE, names=["tweet"])
make_vector = CountVectorizer(analyzer = "word", max_features=10000)
transf_all=make_vector.fit(tweets)
keywords= transf_all.transform(new_features)
tfidf = TfidfTransformer().fit(keywords)
X = tfidf.transform(keywords)
print X[0:2]
```

The output result which is a sparse matrix can be represented as the following based on the frequency of each word in the whole context the first 2 tweets represented, every tweet is represented as a row in a sparse matrix which can be seen as a long array to the length of the vocabulary size, so that only words of that tweet have non-zero frequency, for example the first tweet has 8 words which has non-zero values corresponding to their frequencies in the whole context:

|           |                |
|-----------|----------------|
| (0, 9385) | 0.333212609335 |
| (0, 5045) | 0.521059090768 |
| (0, 4184) | 0.393867037204 |
| (0, 2108) | 0.117261988326 |
| (0, 1419) | 0.335122632708 |
| (0, 1396) | 0.335122632708 |
| (0, 1374) | 0.335122632708 |
| (0, 408)  | 0.334161079641 |
| (1, 8376) | 0.449906112839 |
| (1,7065)  | 0.50331079035  |
| (1, 6336) | 0.50331079035  |

|           |                |
|-----------|----------------|
| (1, 3241) | 0.528867013672 |
| (1,2108)  | 0.106022017118 |
| (2, 8495) | 0.247008510761 |
| (2, 8214) | 0.380496851262 |
| (2, 7692) | 0.379926609191 |
| (2, 6901) | 0.382812276721 |
| (2,5213)  | 0.61263115134  |
| (2,2116)  | 0.319185899645 |
| (2, 2108) | 0.161625827414 |
| :         | :              |

There exist many different approaches or methods used in preprocessing and vectorizing the textual data, these two steps, included in the process of "feature engineering", they are typically the most time consuming parts of building a predictive pipeline, but they are very important and require some experience because they can improve the accuracy in finding the data patterns.

### 5.3 fastFM

fastFM is a library for factorization machines in Python, the core is in C and can be used stand alone. It has a user interface in Python with the name of Factorization Machines (FM) which are only used in a narrow range of applications and are not part of the standard toolbox of machine learning models. (Bayer, 2015) FMs are known as successful approaches for recommender systems they are a general model to deal with sparse and high dimensional features. fastFM implementation provides easy access to many solver methods for learning process and supports regression, classification and ranking tasks. This implementation has the potential to improve our understanding of the FM model and drive new development. Keywords: Matrix Factorization, Recommender Systems, MCMC. At the beginning the goal of the thesis was to use the fastFM for matrix factorization based machine learning models. Factorization Machines are capable to express many different latent factor models that are widely used for collaborative filtering tasks, mainly we wanted to extract the latent features of the context. In this case it was supposed to have the features as like the current case (all the words before the second hashtag and the first hashtag) the output value of this factorization machine was expected to be the second hashtags. Although there was a problem we face in this regard which was the fact that this algorithm was implemented for the binary classification methods so in our case where we have huge number of classes (second hashtags) it was impossible to succeed, although many different methods were examined to solve this issue.

One of the methods we tried was to use sklearn. Multi-class module in python which can decompose the multi-class classification problems into binary classifications. There are two different strategies for this task, One-Vs-The-Rest and One-Vs-One.

**OneVsTheRest** strategy, also known as one-vs-all, is implemented in `OneVsRestClassifier`. In this method the policy is fitting one classifier per class. The class is fitted among all the other classes for each classifier. In addition to its computational efficiency (only `n_classes` classifiers are needed), one advantage of this approach is its interpretability. Because each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy and is a fair default choice.

**OneVsOneClassifier** constructs one classifier for every pair of classes. At prediction time, the class which received the most votes is selected. In cases of a tie (when there are two classes with an equal number of votes), it selects the class with the maximum aggregate classification confidence by summing over the pair-wise classification confidence levels estimated with the based on the binary classifiers.

this method is usually slower than one-vs-the-rest, because it requires to fit  $n\_classes * (n\_classes - 1) / 2$  classifiers, and its complexity is estimated as  $O(n\_classes^2)$ .

However, this method may be advantageous for algorithms such as kernel algorithms which don't scale well with `n_samples`. This is because each individual learning problem only involves a small subset of the data whereas, with one-vs-the-rest, the complete dataset is used `n_classes` times.

Although we tried to apply one-vs-rest to convert the fastFM binary classifier to the multi-class classification case:

```
fm = sgd.FMClassification(n_iter=1000, init_stdev=0.1, rank=2, random_state=123, l2_reg_w=0.0,
                        l2_reg_V=0.0, l2_reg=0.0, step_size=0.1)
OVR= OneVsRestClassifier(fm).fit(X_train,y_train)
```

but the following error at first stopped us for any further progress which has not fixed yet and after contacting with Imanuel Bayer we put effort into solving the error for one about one month and finally he reported this error as a bug for this module<sup>10</sup>:

```
RuntimeError: Cannot clone object FMClassification(init_stdev=0.1, l2_reg=None, l2_reg_V=0, l2_reg_w=0.0, n_iter=1000, random_state=123, rank=2, step_size=0.1), as the constructor does not seem to set parameter l2_reg_V
```

---

10 [.https://github.com/ibayer/fastFM/issues/49](https://github.com/ibayer/fastFM/issues/49)

So the decision was to change the method rather than factorization into much more information retrieval point of view, and pure machine learning which will be described in detail in the following.

## 5.4 Cross-Validation

After the data collection, cleaning and preprocessing is done cross-validation is applied to perform model selection. CV is primarily a method of estimating the predictive performance and accuracy of a machine learning model. Simply one can over-fit the data by adding too many degrees of freedom. For example, in a simple polynomial regression by considering higher order terms it is possible to get better and better fits to the data, although predictions on new data will get worse in many cases as higher order terms are added. One fold of cross-validation inducts partitioning a sample of data into complementary subsets, performing the analysis on one subset (*training set*), and validating the analysis on the other subset (*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

Cross-validation methods can be categorized mainly in 2 sets, **leave-k-out** cross-validation (in which k observations are left out at each step) and **k-fold** cross-validation (where the original sample is randomly partitioned into k subsamples and one is left out in each iteration).

There are 3 main phases in every parametric Supervised ML method, training, validation and testing in which there should provide different sets of data. This is also an important step how we select these data specially when we have a stream data where the predictive behavior of the hashtags is very sensitive to time. The best method is to split these data randomly this way we make sure about the bias-variance trade-off which can be a confusing issue in machine learning that we should be carefully consider it in every step evaluation.

Important fact about this step is that the proportion of data dedicated to each set, has a great effect in the accuracy and final output result, means that There is no fixed rule about this proportion, it depends on the complexity of the situation (Application) and how many independent parameters you have chosen. Also it can be depending on the size of the data set in huge datasets like our case it is better to consider enough portion to test and validation sets but in case of small data sets it usually happens that more than 80% of the data is considered as the training set. increasing the training accuracy and reduce the training error but it doesn't mean that we have successfully created a successful generalized model but also we should also try to check if the model is relatively acting good with unseen testing data.

In this thesis we also have these 3 sets:

From the beginning we separate a validation set which is a very important set to evaluate the model performance, in this thesis we intent to examine different ML methods on this dataset and compare their performance, so it even makes the efficiency of the validation set much more prominent. So from the beginning a subset of the whole dataset will be discarded to be dedicated to the evaluation set, in our case after trying 4 different portions among 25%, 20%, 15% and 10% the best one was 15%, so by this 15% randomly split subset from the data we mainly focus on selecting among collection of ML methods in candidates in this thesis also to tuning the parameters of the selected algorithms. For example, based on this subset we could select 3 or 4 methods among a set of algorithms such as {LDA, QDA, DecisionTree, RandomForest, SVM (linear & non-linear kernel), KNN and Naive Bayes} and also to tune their parameters.

Training set has the majority volume of the data, here we have a pairs of tweets features and the corresponding outputs in all the evaluations we consider 75% of the remaining data to be defined as the training sets. 2 other portions (80% and 90%) were also tried but the best results both in training and generalization was achieved by 75% portion to training and 25% to testing. This set is used to build up our prediction algorithms. Each type of algorithm has its own parameter options (the number of layers in a Neural Network, the number of trees in a Random Forest, etc), algorithms will tune their selves to the mutations in training data sets.

Testing phase is included in parametric approaches in non-parametric approaches we consider the validation step as the test step. So far we have found the preferred prediction algorithms but we we would like to examine them on the completely unseen real-world data. Then we check the performance if it was not satisfactory then we should start from the training set and re-tune the models. This set also represent if there occurred any over-fitting during the training set in that case the training error would be incredibly low but while we try the testing set then we face a huge amount of error that means we are not learning and modeling the data but we are simulating one by one of the input sample data.

#### ***5.4.1 Cross-validation on stream data***

The main concept of Cross-Validation already explained, the well-known algorithm in the machine learning is to randomly select the samples out of the dataset and split it into training, validation and testing sets, more precisely the steps can be described as the following



- 1.Split randomly data in train and test set.
- 2.Focus on train set and split it again randomly in chunks (called folds).
- 3.In case of total 5 folds; training is on 4 of them, testing is applied on the 5th.
- 4.Redo third step for 5 times to get 5 accuracy measures on 5 districts folds.
- 5.Estimate the accuracy average of folds, represent the model performance

The problem is that this method (which can be easily implemented by usual built-in Scikit functions) cannot be applied on the stream time-series data which is the main consideration of this thesis. Because in the stream data cases like tweets, data cannot be shuffled randomly, as we lose its natural order, which cause matters. So the method we followed to apply proper cross-validation on Expo data set was to consider the timestamps in this step.

One possible solution can be the following structures<sup>11</sup>:

- 1.Split data in train and test set given a Date (i.e. test set is tweets on 12 April 2015).
- 2.Split train set (i.e. tweets from one week before 12 April 2015) in for example 10 **consecutive time folds**.
- 3.Then, in order not to lose the time information, perform the following steps:
- 4.Train on fold 1 → Test on fold 2
- 5.Train on fold 1+2 → Test on fold 3
- 6.Train on fold 1+2+3 → Test on fold 4
- 7.Train on fold 1+2+3+4 → Test on fold 5
- 8.Train on fold 1+2+3+4+5 → Test on fold 6
- 9.Train on fold 1+2+3+4+5+6 → Test on fold 7
- 10.Train on fold 1+2+3+4+5+6+7 → Test on fold 8
- 11.Train on fold 1+2+3+4+5+6+7+8 → Test on fold 9
- 12.Train on fold 1+2+3+4+5+6+7+8+9 → Test on fold 10
- 13.Compute the average of the accuracies of 9 test folds (number of folds – 1)

To know more in detail, the code of the above algorithm which we have also used can be found in appendix.

---

11 [.http://francescopochetti.com/pythonic-cross-validation-time-series-pandas-scikit-learn/](http://francescopochetti.com/pythonic-cross-validation-time-series-pandas-scikit-learn/)

# chapter 6

## 6. Testing and Evaluation

As already explained in detail this thesis is supposed to work on the dataset of tweets reported in Expo2015 as an on-line stream data set, with the aim of predicting the second hashtags in tweets. The problem is that at the moment twitter is not suggesting a very time-precise hashtag with regard to the current top news and events, so the main task in this thesis was to represent that by applying smaller time windows and exploiting proper machine learning techniques it is possible to time-precise hashtag prediction. For this purpose, there have been different methods used in different steps of the work and after examining and evaluating the results, the best one has been picked to represent the final output.

For example in preprocessing many different libraries and modules in Python have been used like different libraries in stemming and lemmatization, for example after comparisons among ordinary stemming function to extract the stem of a word in python and the language specified ProterStemmer from gensim.parsing the second method seems to be more adequate in our case, also two different libraries implemented in Python (one from Google and the other one from MIT) we examined on the data to detect the language of the tweets and finally the languagedetect API appear to be more powerful in our application.

In this chapter we explain more in detail about different testing applied on this work, rather a previously baseline which has been done on the same database and with the same objective of predicting the second hashtags, but using different methods, this work has successfully accomplished a task to show the effect of proper time windows on prediction of the second hashtags of time stream of micro-bloggers like tweets.

The method which has been used is co-occurrence and the structure is mainly based on the statistical knowledge of the data rather than deeper semantical aspects of the context. But this research apparently is proving the effect of considering the time-precise predictive behavior of the data and improve the final accuracy in the hashtags predictions.

## 6.1 Baseline

This thesis rather than the main hypothesis to show the daily time window can be previously on the same data collection (Expo2015), a project with the same goal of predicting the second hashtags, has been done which we consider as the baseline of this thesis, and try to improve the results and achieve higher precision and lower error rate regarding to what has been successfully achieved. The previous work has been represented a comparison among that co-occurrence method can the Top-K method, currently approach used in twitter.

### 6.1.1 Co-occurrence

A text document can be converted into the sequences of word/word co-occurrence as well as their frequencies are kept. It has been one of the most frequently used methods in text analysis. A good performance is achieved on the experimental data set by using this system (11). This can be seen as sequence analysis system of context used in text analysis and classification problems, which can automatically solve the high dimensionality problem for large data set. This method can be seen as pure statistical investigation and analysis of the textual data. co-occurrence of two words, can be gained by context-vector, this approach is widely used in statistical NLP. To create it we need a corpus.

For every word in the context we define this vector with a fixed length to the number of words in the document, the context vector for each word represent number of times other words have co-occurred with the current word and capture the relationship among words (10), there is a concept called window of word, in fact in a window of words, is possible to find other words occurred with the current word and gain their corresponding value in the context vector. So the Co-occurrence matrix out of these vectors is a matrix were  $N$  is the number of words in the document, the correlation among 2 words in the document is represented with the co-occurrence weight of these two words.

For example, imagine we have a very simple corpus as the following:

corpus: {A D C E A D F E B A C E D}

In case of window size 2 where we only want to check the correlation among two words:

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 0 | 1 | 3 | 2 | 3 |
| B | 1 | 0 | 2 | 2 | 3 |
| C | 3 | 1 | 0 | 2 | 2 |
| D | 2 | 0 | 2 | 0 | 4 |
| E | 3 | 1 | 2 | 4 | 0 |

This baseline is considering only the relation among the hashtags in the tweet, it focusses on the order of occurrence of the hashtags precisely to predict the second hashtags by different time-window length applied on the data. Afterward the result has been compared to the other method, Top-K hashtag which is a predicting the most occurring hashtag with regards to the first hashtag. The steps that have been followed on this method have been the following:

### **Data processing**

Creating a table by looking at the order of appearance of the hashtag in the twitter text so that for each tweet we take its timestamp, the first and the second hashtags in this base line there has not been any preprocessing of the data which can significantly enhance the final performance of the ML model:

$H = \langle ts, h1, h2 \rangle$

i.e. <timestamp, 1st hashtag, 2nd hashtag>

### **Experiment**

after splitting the dataset into test and training data, for every hour  $t = 1, \dots, T$  in the dataset first we start the training phase to properly fit the model the feature set is only consist of the first hashtags and the output predicted value is the second hashtag:

train dataset:

$X = \{h1 \in H : ts \in [t - w : t]\}$

$y = \{h2 \in H : ts \in [t - w : t]\}$

test dataset:

$X = \{h1 \in H : ts \in [t]\}$

$y = \{h2 \in H : ts \in [t]\}$

Where  $w$  is the train window size (e.g. hours).

### **Prediction mechanisms**

In this baseline research there have been applied different time windows and compared the output error rates of each method with regards to different window lengths. In the following the length of the time windows and the used algorithms

**Evaluate time windows:**

w = 1 one hour;

w = 24 one day;

w = 24 \* 7 one week;

w = inf all data;

**Let co-occurrence do:**

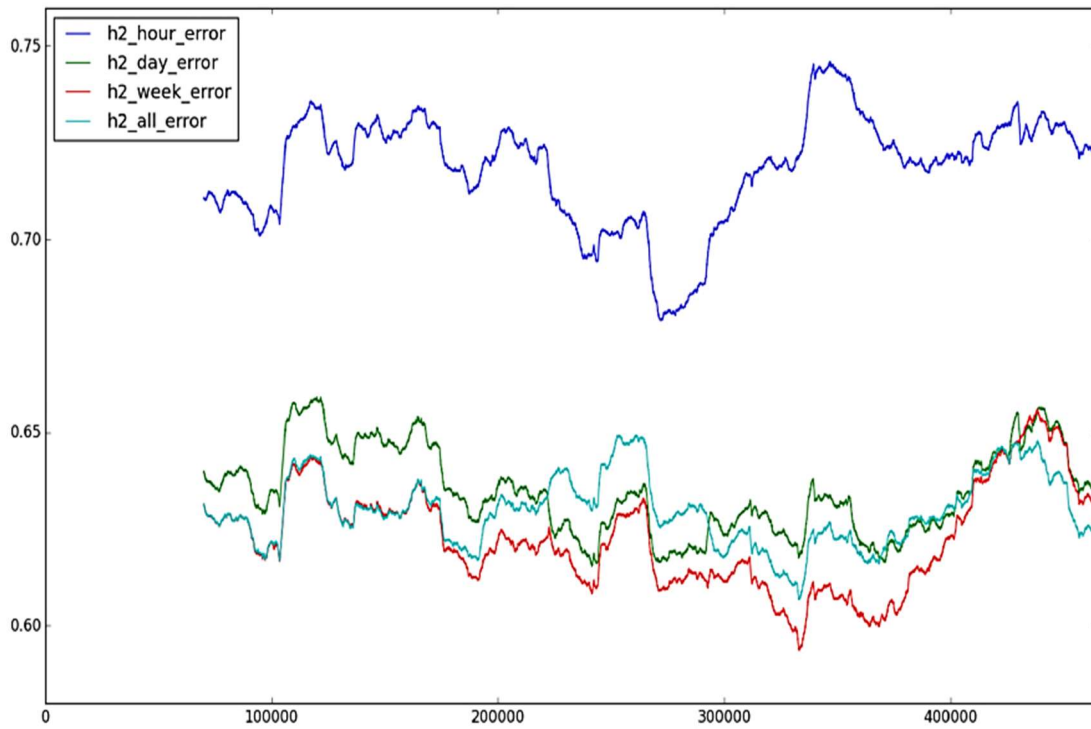
given train data X , needs to predict h2  
if h1 is in X : predict the most co-occurring h2  
else: predict the most occurring h2.

**Let top-tag do:**

given train data X , needs to predict h2  
predict the most occurring h2 regardless of h1  
(can predict h2 == h1).

### 6.1.2 Results

In the following plots the X axis is the tweet count over time and the Y axis represent the prediction error rate. In the first plot the data has been recorded from 23rd June to September 1st. Moving average of 50000 tweets.



*Figure 21 baseline (co-occurrence) prediction errors*

The second plot is the same experiment excluding the top-10 hashtag from the evaluation (i.e. exclude tweets with h2 in the top-10).

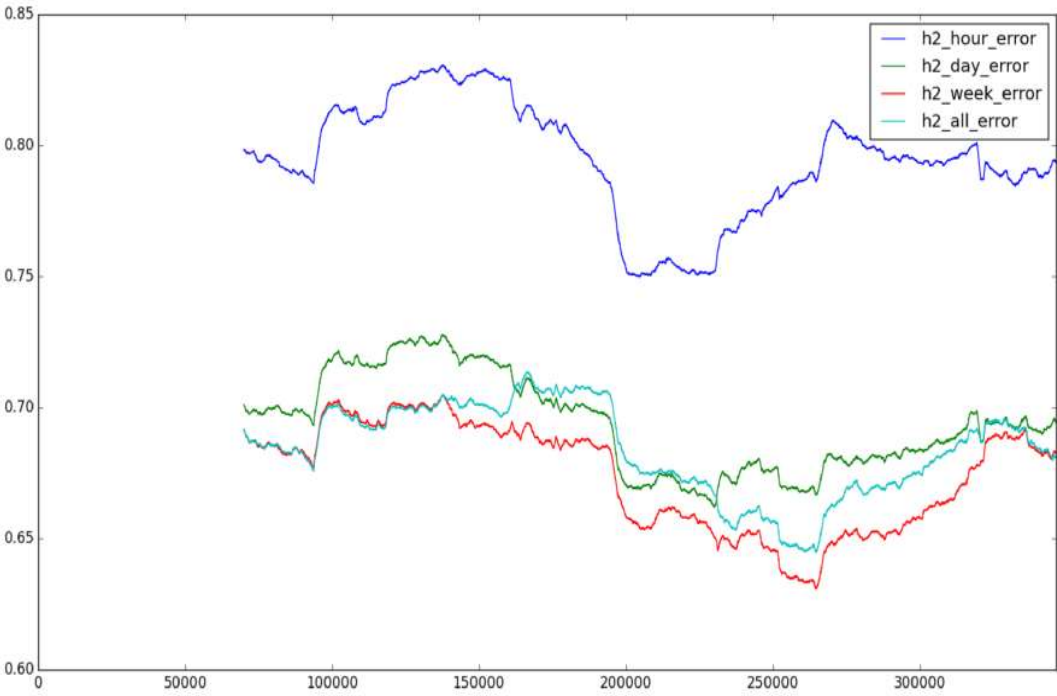


Figure 22 Top-K prediction error rate

The last figure is putting all in one shell and make the comparison easier among these 2 methods, all hashtags are included (no removal of top 10). Note that top-tag-week and top-tag-all overlap:

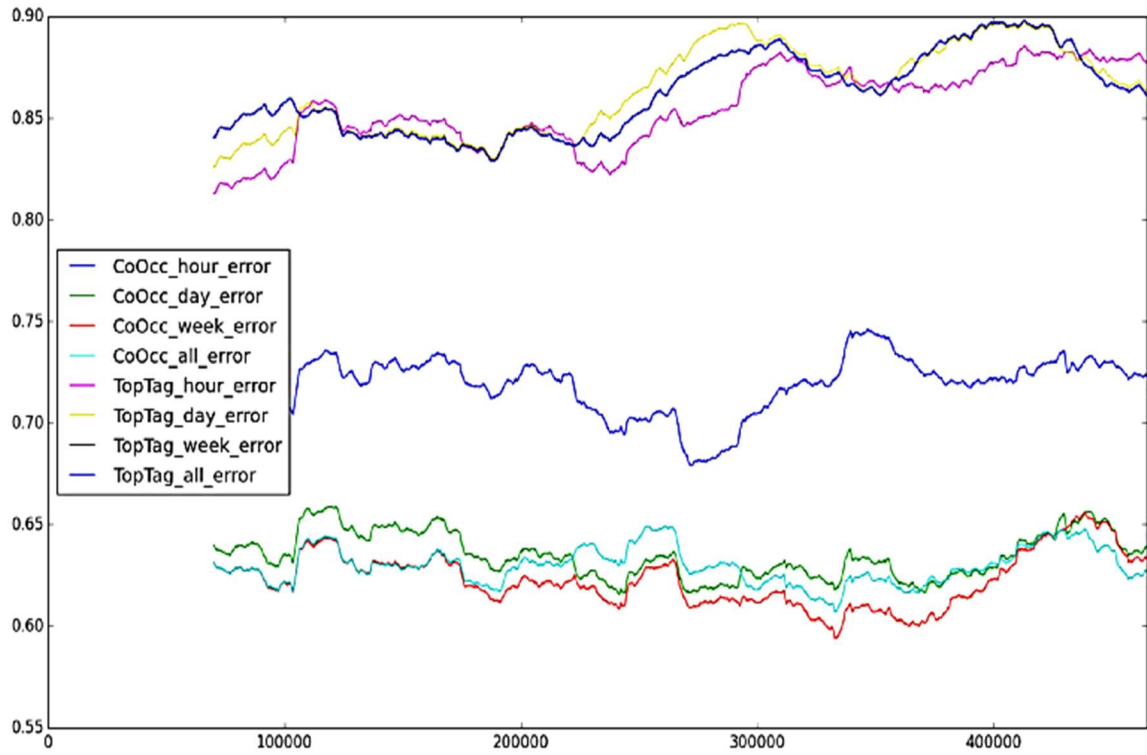


Figure 23 comparison among top-k and co-occurrence



### 6.1.3 Conclusion

Co-occurrence is outperforming top-tag baseline. The experiment was on a limited scenario. The data was just containing the first two hashtags present in the tweets. The co-occurrence simple approach performs generally better when trained on a week period only.

In general, a good job was done in the mentioned baseline and it successfully achieved better result with regards to the Top-K- hashtag method. It correctly reduces the errors up to 25% which can be considered as a perfect result although this method did not meet the main hypothesis of this thesis which said smaller time-window lengths can enhance the precision of the second hashtags prediction. Although from machine learning and data analysis point of view this pure statistical method can be improved much more accurately by including vital preprocessing and feature engineering steps and applying different ML approaches to get more deep in the semantical layers of the data which enable us to act more intelligently and more properly in different general applications. In this thesis we tried to meet the main hypothesis which is the slower window length can increase the precision of the final result in predicting the second hashtag, also the method which is represented has been tried to exploit the latest techniques available in text-processing

## 6.2 Experimental Tests

Many different techniques in different steps of the thesis have been tested and the best one has been selected by the best we mean the one that has appeared to be efficient in enhancing the final result. From the very beginning we start on a small subset of data with 50000 tweets, different preprocessing was applied on the data and bag-of-words to numerically vectoring the features of the data, then we try to fit different ML algorithms (SVM & KNN) and compare the results, at this step the error was incredibly high because the number of features with regards to the number of predicted values (second hashtags) where was not in standard form! But the idea to apply dimension reduction on the second hashtags and applying Neural Networks to involve in the deeper layers of the data helps to somehow cluster the huge number of predicted values into a sets of similar words with the intended predicted hashtag, in fact after this step rather than having only single words as the predicted values we had an embedding vector corresponding to the output values. By this we claim that the model was learning to predict intended hashtag and also the similar words to this word. Considering only the top-10 similar words in the embedding vectors we had a great progress in the output result accuracy. This has been represented briefly in the following table. We test 2 different embedding approaches, Word2vec and Glove the result was close as you can see in this table.

*Table 3 effect of dimension-reduction on error rate*

|                                   | <i>KNN</i> | <i>SVM</i> | <i>Bayesian</i> |
|-----------------------------------|------------|------------|-----------------|
| <i>Primary Simple Case</i>        | 0.88733    | 0.73149    | 0.71062         |
| <i>Word2vec Embedding applied</i> | 0.71861    | 0.53980    | 0.48228         |
| <i>GloVe Embedding applied</i>    | 0.74652    | 0.55193    | 0.50941         |

So we can see that the better result is achieved by Word2vec and we consider it as the default embedding technique, also Naive Bayesian seems to be a better approach among other ML methods then also this method is selected as the default ML algorithm for prediction task.

Next to eliminate the error rate and increase the precision of the final prediction we put more effort on the preprocessing of the data although we already select the best possible libraries for this purpose but the fact of having tweets in different languages was remained with lower emphasis in preprocessing. we usually fix all steps of the preprocessing, as an exception here things go on the other way around, but the result once again yields the importance of the preprocessing on the data set. By applying customized and language specified stop-words and stemming techniques we have achieved even better result. The main approach which was the Languagedetect API in Python had a premium service which provide for free for one month of academic use for this thesis and I was significantly helpful. At this step we consider two main cases the first one was to only consider the English tweets and eliminate the rest, and the second scenario was to identify English and Italian Tweets by this API and apply language specified preprocessing techniques. The following table is representing the error rate with word2vec embedding method

*Table 4 effect of specialized preprocessing on error rate*

|  | <i>KNN</i> | <i>SVM</i> | <i>Bayesian</i> |
|--|------------|------------|-----------------|
| <i>Considering Only English Tweets</i> | 0.53301    | 0.34376    | 0.27992         |
| <i>Separate English/Italian Tweets</i> | 0.61967    | 0.44701    | 0.40643         |

Although we can find a better result in the case where we are only considering the English tweets and eliminate the rest, it is not the desired case with regards to the nature of the data and the goal of the thesis. The main problem of this strategy is that we are removing many part of the dataset which can convey huge amount of information and it is a principle in cleaning data state. This way we consider the second strategy as the default strategy which is detecting Italian and English tweets and apply language-specific techniques on each set separately, but tweets from both of these languages are involved in the dataset. There exist tweets in other languages

but their portion with regards to Italian/English tweets is really negligible which makes problem in training step and fitting the ML model.

In the following figure which is plotting different ML method daily errors based on the day on which that tweet has been reported we represent a general case where the testing and training set has been selected from the whole dataset but only to check the results we separate the testing data based on the timestamps day by day. In this figure there exists fluctuations in all the methods, but one can also find some peaks or widely fluctuations in some points, the reason for this problem can be seen as the fact that in these days the system which recorded the tweets were temporarily off for several hours on those days so there have been very few number of tweets reported on that day which means that there have not been enough samples to be examined. These points are also represented in the frequency of tweets plot in the previous chapter. One of the days is the 153<sup>rd</sup> day of the Expo where there are only 128 tweets recorded and even these tweets are not all included in the intended valid tweet set it was also seen in several other days like days 65,116,149 etc. but in these days at least we had 1000 tweets recorded on those days which cause smoother mutations. The main point is that there is almost a moderate behavior in all the methods and among all the considered methods Naive Bayesian is outperforming others, although SVM with non-linear kernel is also representing a moderately good performance almost close to the NB method but among these 3 methods KNN is not representing a good performance in predicting the second hashtags. In some days we can also see the lower errors which can be defined as the cases where we had great events which reduce the variation of reported second hashtags and so we have been succeed to learn more efficiently the data pattern of that days.

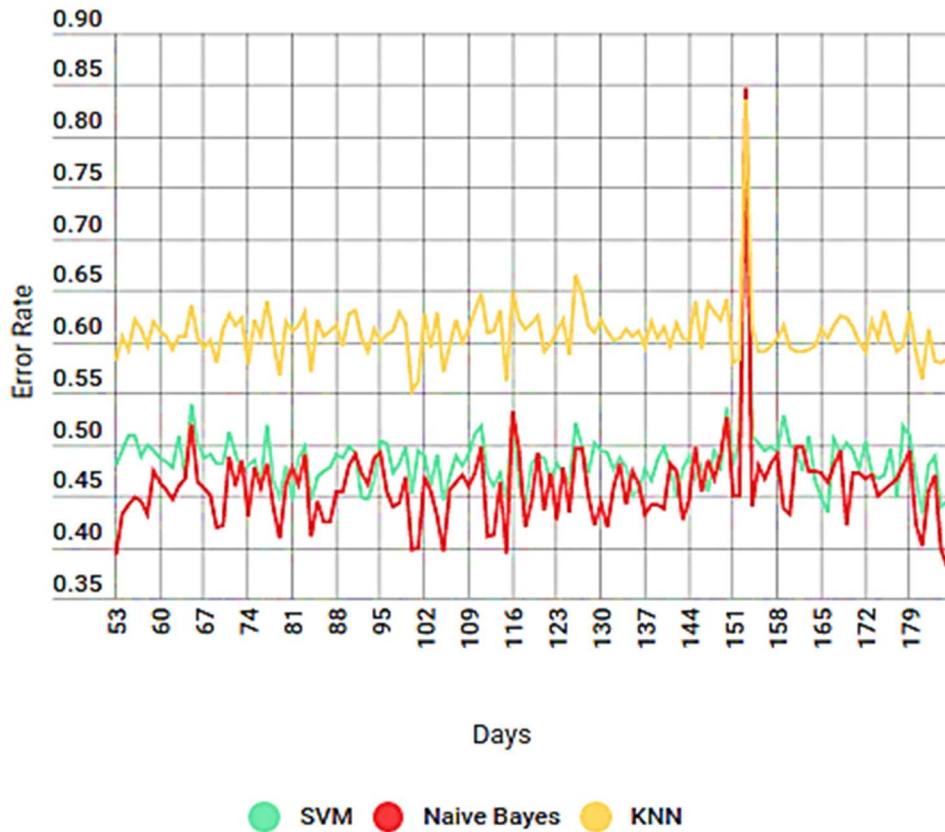


Figure 24 Performance comparison among different methods in machine learning

As the main hypothesis of this thesis have been defined as applying daily time window can enhance the accuracy of the prediction task because it can be closer to the predictive pattern in the data (based on the daily events in Expo and relevant tweets with specific hashtags and keywords). This fact motivates us to apply proper window time lengths so that the model can be updated regularly in different periods. In the bellow figure we can clearly notice the effect of having time window on the final accuracy result. Rather than the effect of the windows it can be seen as the effect of updating the semantical relations among words in the embedding step and refitting the model based on the new data patterns of each segment. There is another policy in updating the ML models with stream data can be to update the system every moment that new feature appears. The second approach can be more accurate but is not an optimal way in our case, the process we are following have 2 main section of preprocessing the data and ML/Dimension Reduction phase and huge number of stream of tweets which at each time stamp there might be tweets containing new feature (new topic or new event), it can be really time consuming to rebuild all the previously done steps from the scratch every time. The following figure represent the daily MSE error rate estimated in different time windows we consider all the mentioned default cases (using word2vec embedding, Bayesian algorithm and applying second language specified policy already explained), fluctuation behavior in all the cases but it seems like fluctuations in smaller windows are little bit slightly and

moderate. It shows the daily lengths of the windows had successfully reduced the error on average around 17% to 19% and weekly updating has achieved the reduction of 11%-13% on average. Those sharp jumps still represent the poor collection of tweets in those days, but it is not like we have the same fluctuating patterns in all the cases rather we can see the points which can claim the importance of setting a proper time window length by understanding how the nature of data is modifying by time. For example, in day 116 we have the lowest jump in daily windows the reason can be efficiency of proper window length in the learning process in our case.

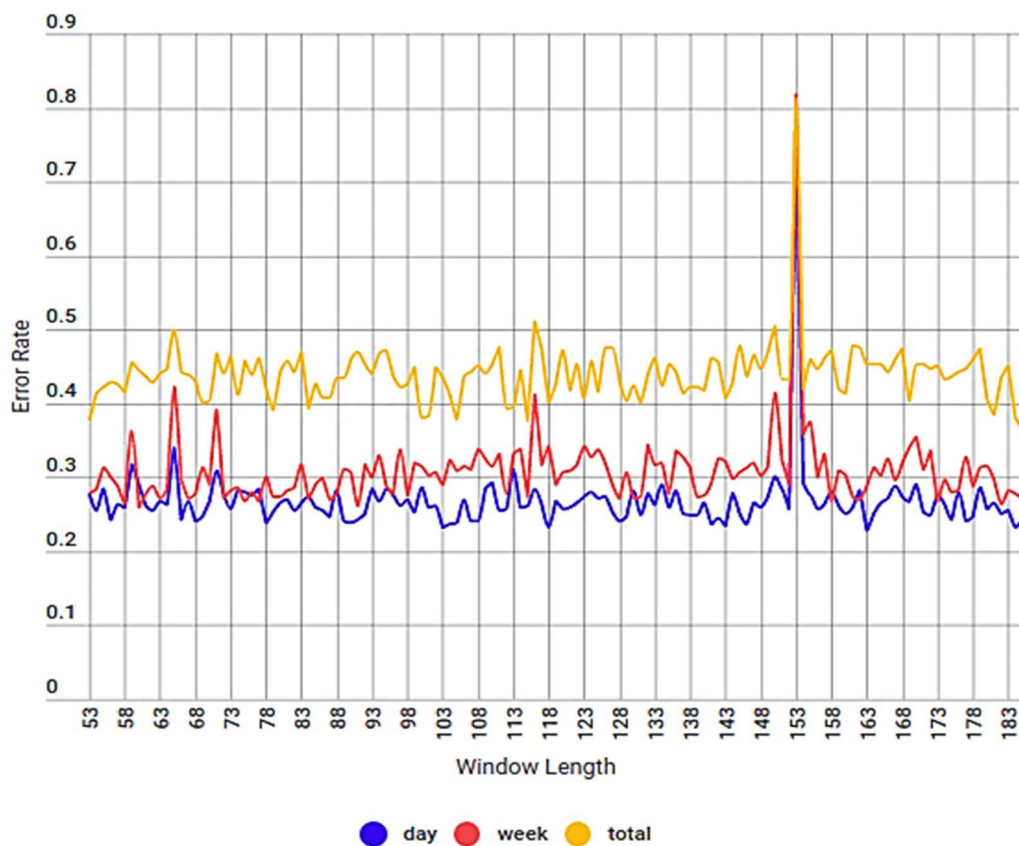


Figure 25 time window effect on the second hashtag prediction

# Chapter 7

## 7. Conclusion

Summing up, we are now going to have an overview about problems faced, how we solved them but above all we are going to focus on results, conclusions drawn and possible future developments.

### 7.1 Problem and adopted solutions

The problem we tried to solve in this thesis work was to find a way for efficient prediction in on-line micro-blogger. The dataset we work on was a set of tweets collected in Expo2015 consist of 2 million of tweets, more specifically the goal was to predict the time-precise second hashtags of the tweets given the words occurred before the second hashtags. In particular, our work was consisting of 3 main sections, first part is focused on preparing and cleaning data to be converted into feature vectors and the second section is concentrated on prediction task using machine learning algorithms and increasing the accuracy by dimensionality reduction and ML techniques. The last section which is th answer to the main hypothesis of the work, is to apply different time windows and compare the prediction result. The expectation was to see a better result in smaller daily time window. We compare weekly and daily time-windows together and the case with the total data with no time-window at this step.

Already there have been a baseline done on these tweets with co-occurrence method and represent the results comparing to Top-K method. It can be considered as a purely statistical method which tries to predict the second hashtags based on its co-occurrence with the first hashtag, although this method has been outperform the Top-K method also after applying the time-windows but, it could not achieve the

main goal which was getting to better result in smaller time-window, rather the result with this method claim that the weekly window length outperform smaller window lengths, the reason can be seen in two points of view, first this method is only considering the co-occur of 2 hashtags as the input features for the prediction and does not have a huge number of degree of freedoms which means it is not a flexible method with regards to the small variations in the data behavior so that is why it act better on the bigger time windows which are not affected by small changes in the data behavior happens for a short time. The other fact about this method is that it does not consider other words in a tweet which can provide vital information for predicting the second hashtag and this part of information is missed and this affect the final prediction result.

So to find the answer to the main hypothesis we have followed other methods with more emphasis on feature engineering and applying recent methods based on neural networks to get deep in syntactical layers of features in tweets in order to provide more detailed useful information for better prediction result. From the very beginning after cleaning the data by several common methods and providing a proper feature representation of tweets. We test different ML approaches by these feature sets, but the result was not satisfactory! To solve this problem and to improve the precisions in the result of these methods, we followed 2 remedies, the first solution was mainly focused on improving the preprocessing step quality and concentrate on feature engineering. For the first solution rather than common approaches, in order to improve the quality of preprocessing by extracting semantical information in deeper layers of the context (by applying a 2-layer feed-forward Neural Networkin embedding techniques) or to find the language specific stem of each tweet (language detect API and Gensim Stemmer module) and to use more reliable methods with strong datasets in lexical and morphological steps (NLTK, BeautifulSoup). After exploiting the first solution we had a great progress in the final out put result we increase the average precision of the prediction up to 30% at this step which guaranteed the importance of preprocessing and feature engineering steps in machine learning techniques.

The second solution was to consider the fact that tweets topics might be in high percentage related to the popular events of the day which can affect the predicted hashtags in only short period of time(day) this means that the second hashtags prediction behavior could be affected by time to make sure about it we needed to apply different window lengths to investigate this fact on different time-precise segments of the data, we consider day and week as the window lengths, after testing the data once again the precision was improve this time we successfully meet the goal and show that the best case in results was achieved by the daily window length. At this step we had increase the precision up to 20%. We also have a great progress in weekly window length which demonstrate that the hypothesis is correct.

## 7.2 limitations

The method we followed is represented almost good performance based on the goal we set for this thesis, which was to prove that the daily time window can act more accurate in predicting the second hashtags. Although the final result still can be improved, among different machine learning methods examined on the dataset the best result was achieved by Naive Bayesian, even t-in the best case the result was not any better than 24% of errors in hashtag prediction. In Machine Learning amount of the error is evaluating with respect to the applications or the nature of the dataset or the problem we defined. But in this application work it seems to be possibly improved by working more on the ML or Feature Engineering techniques.

Also in this thesis there has not been any considerations for other languages tweets in general case while processing the micro-bloggers there might be tweets from other languages which need to be processed in the same way that it has been done in this thesis although there was two obstacles on this purpose in our case the first one is that we had not enough number of tweets in other languages which could make difficulties in training steps as we need enough number of samples to fit a model in ML, the second problem is more general case which is we may not have an abundant libraries and modules as we have in English for different NLP and preprocessing tasks or in some cases that one can find some libraries they might be usually act not as that strong that we expect, this can also affect the final performance of any ML task.

## 7.3 Future Works

considering the current project work as a baseline we can define a future work purpose to focus more on embedding or dimensionality reduction aspects of the work in this thesis, we could reduce the great amount of the final error only by using a 2-layer feed-forward neural networks too group-wise the output values. In case one get deeper in the layer it is expected to have a better result in the final performance, also it might be interesting if we apply this technique on input features so that instead of only having a frequency as the value of each feature we can have a vector instead. Although this might be just easy to say in theory and practically there might be obstacles like how to deal with standard acceptable feature formats in ML algorithm or even from optimization point of view it takes so much time to convert the features in the desired format. It might be challenging but really interesting.



## References

1. *A fast and simple algorithm for training neural probabilistic*. Andriy Mnih, Yee Whye Teh. s.l. : Appearing in Proceedings of the 29 th International Conference, 2012.
2. *Tailoring Continuous Word Representations for Dependency Parsing*. Mohit Bansal, Kevin Gimpel, Karen Livescu. s.l. : 52nd Annual Meeting of the Association for Computational Linguistic, 2014, pp. 810-823.
3. *Natural Language Processing (Almost) from Scratch*. Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu and Pavel Kuk ' sa. 2011, pp. 2493-2537.
4. *A Neural Probabilistic Language Model*. Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin. s.l. : Journal of Machine Learning Research, 2003, pp. 1137-1350.
5. *Label updating to avoid point-shaped obstacles in fixed model*. F. Rostamabadi, M. Ghodsi. s.l. : IEEE, 2006.
6. *Reducing the Dimensionality of*. Salakhutdinov, G. E. Hinton\* R. R. 2006.
7. Dr. S. Vijayarani, J. Ilamathi. *Preprocessing Techniques for Text Mining- An Overview*. 2014. 2249-5789.
8. *Natural Language Processing (Almost) from Scratch*. Collobert, Ronan, Weston, Bttou. 2011.
9. Data preprocessing. *www.cs.ccsu.edu*. [Online] [Cited: 01 06, 2016.]
10. *WORD CO-OCCURRENCE AND THEORY OF MEANING*. Lancia, Franco. 2005.
11. chen, Guan-Bin and Hung Yu Kao. *Word Co-Occurrence Augmented Topic Model in Short Text*. 2007. pp. 45-64.
12. co-occurrence. [Online] 2015. *www.soc.ucsb.edu*.
13. *Efficient Estimation of Word Representations in Vector Space*. Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2014.
14. —. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. 2014.
15. *An Introduction to Feature Extraction*. Guyon, Isabelle, and André Elisseeff. 2004.
16. features-of-data-and-information. *https://www.ukessays.com*. [Online] [Cited: 10 24, 2015.]
17. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. Joachims, Thorsten.

18. Gareth James, Daniela Witten , Trevor Hastie. *an introduction to statistical learning*. s.l. : Springer, 2013. 978-1-4614-7138-7.
19. Machine Learnin vs NLP . [www.lexalitics.com](http://www.lexalitics.com). [Online] 02 08, 2012. [Cited: 01 19, 2016.]
20. *A Comparison of Classifiers and Document Representations for the Routing Problem*. H. Schutze, D.A. Hull, and J.O. Pedersen. s.l. : Proc. 18th Int'l Conf, 1995.
21. Ghaffari, Parsa. Text Aalysis. [www.kdnuggets.com](http://www.kdnuggets.com). [Online] 01 2015. [Cited: 10 15, 2015.]
22. Khan, Safdar Sardar, and Divakar Singh. *an Effective Supervised Steamed Text Classification Approach for mining positive and negative examples*. 2013. pp. 24-29.
23. *text classification based on labled-LDA model*. Li, W., Sun, L. and and Zhang, D. s.l. : Chinese Jornal of Computer, pp. 620-627.
24. Brownlee, Jason. Implement K/Nearest Neibors in Python. [www.machinelearningmastery.com](http://www.machinelearningmastery.com). [Online] 09 12, 2014. [Cited: 04 23, 2016.]
25. *charecterizing microblogs with topic models*. Ramage, D., Dumais, S. T., & Liebling, D. J. s.l. : AAI conference on weblog and social media, 2010.
26. *short and tweet: experiment on recomending content from information streams*. Chen, J., Nairn, R., Nelson, L., Bernstein, M., & Chi, E. s.l. : SIGCHI conference on Human Factors in Computing Systems, 2010. pp. 1185-1194.
27. *Natural Language Processing*. A. Chopra, A. Prashar, C.Sain. s.l. : International Journal of Technology Enhancement and Emerging Engineering Research, 2013.
28. *Short Text Similarity with Word Embeddings*. Kenter, Tom, and Maarten De Rijke. 2014.
29. *Deep Learning in Neural Networks*. Schmidhuber, Jurgen. s.l. : elsevier, 2015.
30. *Parsing Natural Scenes and Natural Language with Recursive Neural Networks*. Socher, Richard, Cliff Chiung, -Yu Lin, Andrew Y Ng, and Christopher D Manning. 2014.
31. *Active Learning from Data Stream*. X. Zhu, P. Zhang, X. Lin, and S. Y. s.l. : ICDM 06, 2007.
32. *Undrestanding Bag-of-Words Model: A statistical Framework*. Y. Zhang, R. Jin, Z. Zhou. 2008.
33. *Sampling strategies for bag-of-features inimage classification*. F. Jurie, B.Triggs. 2006. 9th European Confrence on Computer Vision. p. 490/503.
34. *word2vec Parameter Learning Explained*. Xin, Rong. 2013.
35. Lai, Siwei, Kang Liu, Jun Zhao. *Recurrent Convolutional Neural Networks for Text Classification*. 1990.
36. J. Pennington, R.Socher, C. D. Manning. *Glove: Global Vectors for Wrod Representation*. 2014.
37. —. *Glove: Global Vectors for Word Representation*. [www.nlp.stanford.edu](http://www.nlp.stanford.edu). [Online] 2014. [Cited: 09 02, 2016.]
38. Pereira, Fernando. *Machine Learning in Natual Language Processing*. [Online] 2002. [www.web.stanford.edu](http://www.web.stanford.edu).
39. *fastFM: A Library for Factorization Machines*. Bayer, Imanuel. 2015.
40. chopra, Abhimanyu and Sain. *Natural Language processing*. s.l. : INTERNATIONAL JOURNAL OF TECHNOLOGY ENHANCEMENTS AND EMERGING ENGINEERING RESEARCH .

41. *Learning Word Embeddings Efficiently with Noise-Contrastive Estimation*. Mnih, Andriy, and Koray Kavukcuoglu. 2013.
42. Word2vec: Neural Word Embeddings in Java. *deeplearning4j.org*. [Online] [Cited: 05 01, 2016.]
43. Xu, Weiqun, Peijia Li, Yonghong Yan, Content Understanding, and Haidian District. *Distributional representation of Words for short Text Classification*. 2015. pp. 33-38.
44. *Text classification without negative*. X. Jeffrey member. 2008.
45. *One-class classification of text streams with concept drift*. Yang, Z. Zhang. s.l. : University of Queensland Australia, 2008.
46. M. Rajman, R.Besancon. *Text Mining: Natural Language techniques and Text Mining applications*. 2008.

# Appendix

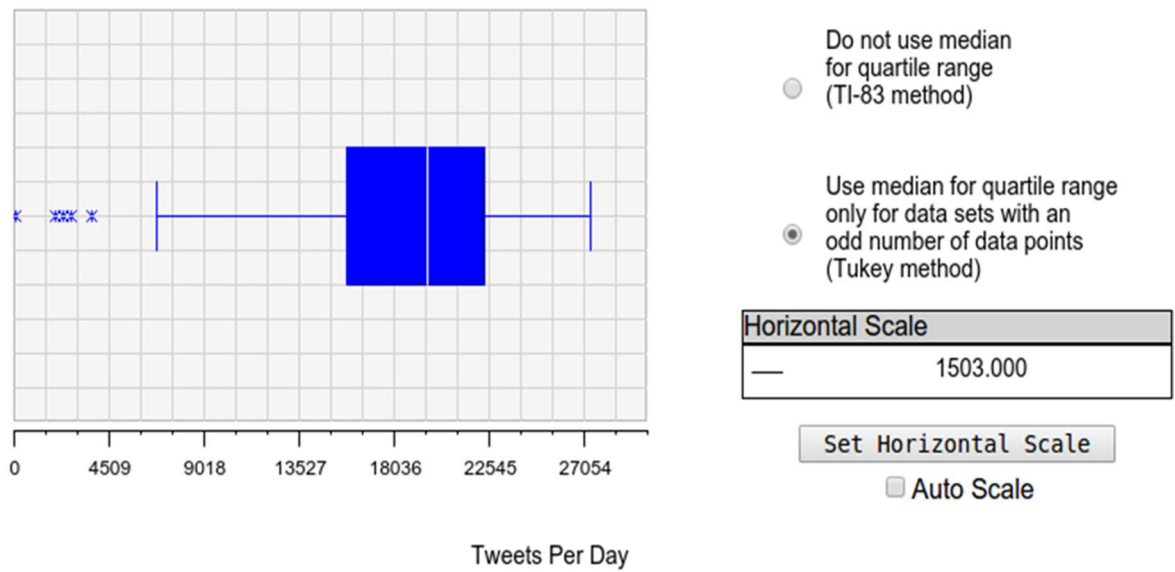


Figure 26 plot-box of tweets per day

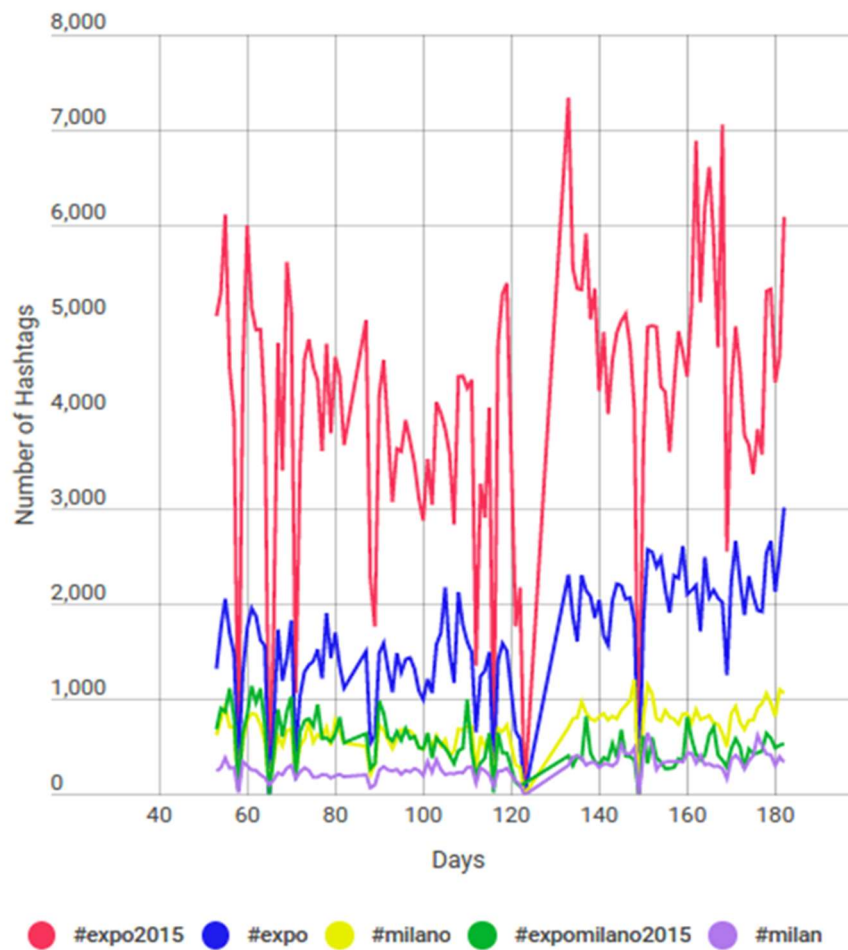


Figure 27 top5 hashtags frequencies

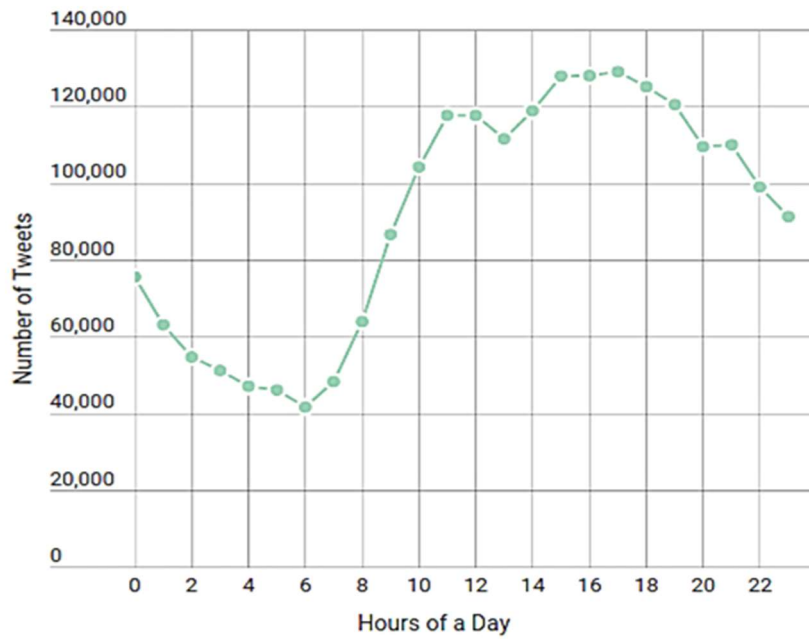


Figure 28 distribution of tweets per hour

**Initial values for outputs before dimension reduction, Y [0:100]:**

|            |     |
|------------|-----|
| (0, 9385)  | 1.0 |
| (1, 7701)  | 1.0 |
| (2, 2108)  | 1.0 |
| (3, 2108)  | 1.0 |
| (4, 2108)  | 1.0 |
| (5, 2108)  | 1.0 |
| (6, 2108)  | 1.0 |
| (7, 2108)  | 1.0 |
| (8, 9385)  | 1.0 |
| (9, 2425)  | 1.0 |
| (10, 9952) | 1.0 |
| (11, 7328) | 1.0 |
| (12, 8469) | 1.0 |
| (13, 8469) | 1.0 |
| (14, 2103) | 1.0 |
| (15, 9952) | 1.0 |
| (16, 2108) | 1.0 |
| (17, 1732) | 1.0 |
| (18, 2195) | 1.0 |
| (19, 2151) | 1.0 |
| (20, 6236) | 1.0 |
| :          | :   |

### Y<sub>i</sub> sample in Word2vec:

[(u'expottimisti', 0.9997639656066895), (u'visita', 0.9997269511222839), (u'padiglione', 0.9995076656341553), (u'tavoliexpo', 0.9994252324104309), (u'expomilano', 0.9994238018989563), (u'rispondono', 0.9994091987609863), (u'news', 0.999385416507721), (u'italy', 0.9993605017662048), (u'milioni', 0.9992889761924744), (u'giorno', 0.9992825388908386), (u'milan', 0.9992722868919373), (u'eventi', 0.9992614984512329), (u'food', 0.9992583394050598), (u'attrazion', 0.9992079138755798), (u'aiuta', 0.9992057085037231), (u'bio', 0.9992033243179321), (u'italia', 0.9991999864578247), (u'casacorriere', 0.9991902112960815), (u'piattaforma', 0.9991848468780518), (u'waiting', 0.999184787273407), (u'eu', 0.9991825819015503), (u'giulianopisapia', 0.999175488948822), (u'spesa', 0.9991652369499207), (u'expomilano2015', 0.9991472363471985), (u'expoidee', 0.9991464614868164), (u'madeinitaly', 0.9991259574890137), (u'padiglioni', 0.9991120100021362), (u'ottobre', 0.9990994334220886), (u'poland', 0.9990965127944946), (u'viaggio', 0.9990962147712708), (u'innovazione', 0.9990894198417664), (u'fun', 0.999079167842865), (u'costa', 0.9990721344947815), (u'cibo', 0.9990658760070801), (u'domani', 0.9990643262863159), (u'colori', 0.9990605115890503), (u'exhibition', 0.9990570545196533), (u'2015', 0.9990461468696594), (u'twitta', 0.9990352988243103), (u'', 0.9990339875221252)]

### Y<sub>i</sub> sample in Glove:

[('eyesdomani', 0.95729623120052354), ('turkey', 0.93607119387162541), ('bizexpo', 0.930557818742072), ('o', 0.92534776719213729), ('baristalife', 0.9145355915846658), ('mostra', 0.90904895924819062), ('bizgewissgroup', 0.90153132882883091), ('volunteerexpo2015', 0.89884831471419868), ('volunteer', 0.89699726356849208), ('sudan', 0.8952444580975466), ('zecchinodoro', 0.89405928213387276), ('swisspavilion', 0.89093092702765475), ('rosatilucit', 0.89077351758095091), ('salute', 0.88831503897192798), ('euexpo2015', 0.88801325002934617), ('bizmusica', 0.88154639108279464), ('unfao', 0.88010911161445493), ('serviziculturali', 0.88010329219622763), ('countdown', 0.87726366700097136), ('ciao', 0.87683778912777166), ('uk', 0.87670820702119945), ('cultura', 0.87608867070691132), ('azerbaijan', 0.87517886793801736), ('usda', 0.86908100213571482), ('exp', 0.86519944320666842), ('cameglobal', 0.85924752977992735), ('gibuti', 0.85890510325905545), ('qatar', 0.85785719953029116), ('feed', 0.8553156123140726), ('orange', 0.85505505913109281), ('ospniguarda', 0.85451505286604645), ('russianpavilion', 0.85398178306195993), ('sera', 0.85236307631435027), ('w', 0.85216168903920586), ('italy', 0.85040123421743419)]

## Cross Validation online stream:

```
def performTimeSeriesCV(X_train, y_train, number_folds, algorithm, parameters):
    """
    Given X_train and y_train (the test set is excluded from the Cross Validation),
    number of folds, the ML algorithm to implement and the parameters to test,
    the function acts based on the following logic: it splits X_train and y_train in a
    number of folds equal to number_folds. Then train on one fold and tests accuracy
    on the consecutive as follows:
    - Train on fold 1, test on 2
    - Train on fold 1-2, test on 3
    - Train on fold 1-2-3, test on 4
    ....
    Returns mean of test accuracies.
    """
    print 'Parameters -----> ', parameters
    print 'Size train set: ', X_train.shape
    k = int(np.floor(float(X_train.shape[0]) / number_folds))
    print 'Size of each fold: ', k
    accuracies = np.zeros(folds-1)
    for i in range(2, number_folds + 1):
        split = float(i-1)/i
        print 'Splitting the first ' + str(i) + ' chunks at ' + str(i-1) + '/' + str(i)
        X = X_train[(k*i)]
        y = y_train[(k*i)]
        print 'Size of train + test: ', X.shape # the size of the dataframe is going to be k*i
        index = int(np.floor(X.shape[0] * split))
        # folds used to train the model
        X_trainFolds = X[:index]
        y_trainFolds = y[:index]
        # fold used to test the model
        X_testFold = X[(index + 1):]
        y_testFold = y[(index + 1):]
        accuracies[i-2] = performClassification(X_trainFolds, y_trainFolds, X_testFolds, y_testFolds,
                                              algorithm, parameters)
        print 'Accuracy on fold ' + str(i) + ': ', acc[i-2]
    # the function returns the mean of the accuracy on the n-1 folds
    return accuracies.mean()
```

## Predicted Values in SVM Method:

```
["art", "padiglionebrasile", "expo2015", "expo2015", "milano", "expo2015",
"expo2015", "expo2015", "todolist", "foodsecurity", "yogyakarta", "naturalhair",
"roma", "roma", "expo", "yogyakarta", "expo2015", "digital", "expottimisti",
"expoidee", "iot", "expo2015", "naturalhair", "regioni",
"semanadelemprededor", "piazzettaer", "milano", "expo2015", "foodsecurity",
"pavilion", "expoidee", "expoidee", "expo2015", "expo2015", "roma",
"expo2015", "experia", "italy", "naturalhair", "expoidee", "expo", "expo2015",
"expomilano", "china", "expoidee", "domani", "expo2015", "puebla",
"expo2015", "polonia"]
```

**First Result on a subset of data (No dimension reduction & no time window)**

```
*****LDA*****  
Linear Discriminant Analysis:  
Accuracy Score:  
0.0261682242991  
*****Decision Tree*****  
Decision Tree accuracy Score:  
0.303738317757  
*****Bayes*****  
Naive Bayes accuracy Score:  
0.309345794393  
*****SVM*****  
SVM Score:  
0.241121495327  
*****Sgd*****  
SGD Prediction:  
[1579 408 2108 ..., 1103 561 2108]  
Score:  
0.214953271028  
*****KNN*****  
KNN Prediction:  
[2108 2108 3579 ..., 2103 2151 2108]  
Score:  
0.172242990654
```