Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Luiza Sayfullina

Reducing Sparsity in Sentiment Analysis Data using Novel Dimensionality Reduction Approaches

Master's Thesis
Espoo, October 15, 2014

Supervisor: Professor Juha Karhunen, Aalto University

Advisor: Yoan Miche, D.Sc. (Tech)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Luiza Sayfullina

Title:
Reducing Sparsity in Sentiment Analysis Data using Novel Dimensionality Reduction Approaches

Date: October 15, 2014 **Pages:** 71

Major: Information and Computer Science **Code:** T-61

Supervisor: Professor Juha Karhunen

Advisor: Yoan Miche, D.Sc. (Tech)

No aspect of our mental life is more important to the quality and meaning of our existence than emotions and sentiments. Recently researches have introduced many Machine Learning approaches to analyse sentiment from public blogs, social networks, *etc.* Due to the sparse and high-dimensional textual datasets one needs Feature Selection before applying classifiers. The scope of my thesis are Dimensionality Reduction techniques for predicting one of the two opposite sentiments, specifically for Polarity Classification.

The greatest challenge for Text Classification problems in general is data sparsity. Especially it is for Bag-of-words model, where the document is represented by the number of occurrences of each term in the vocabulary. Hence it can be hard for a classifier to understand the relationships between all the words in the initial vocabulary when training set is not large enough.

In this thesis I investigate possible steps required to decrease the sparsity: setting the vocabulary, using sentiment dictionaries, choosing data representation and Dimensionality Reduction methods and their underlying strategies. I describe fast and intuitive unsupervised and supervised **tf_idf** scores for Feature Ranking. In addition, Word Clustering algorithm for merging the words with very close semantical meaning is introduced. By clustering semantically close words we decrease the feature space with minimum loss of information compared to Feature Selection, where we simply omit the features.

Polarity Classification problem is investigated on two datasets: SemEval 2013 Twitter Sentiment Analysis and KDD Project Excitement Prediction using Extreme Learning Machine. Best performance for both datasets was achieved by using the proposed Word Clustering and supervised **tf_idf** score with 20 times less features than original vocabulary size.

Keywords: feature selection, text classification, sentiment analysis, tf-idf, word clustering, elm, lars, wordnet

Language: English

Acknowledgements

The thesis was completed in the Department of Information and Computer Science in Aalto University School of Science and funded by D2I TEKES project funding. I am grateful for the opportunity to have my Master's degree in Data Mining and Machine Learning programme, headed by Erkki Oja. Special thanks to Professor Juha Karhunen, who took care about final corrections of my thesis and administrative issues.

My research career has started in Industrial and Environmental Machine Learning research group. I am very grateful to Docent Amaury Lendasse has opened for me the doors to research field, supported with guidance and has found work opportunities for me. My instructor Dr. Yoan Miche gave me the time, support and attention during my Master's thesis writing. Thanks to his good advices, patience and frequent meetings that shaped my standard of research. The members of the research group: Anton, Alexander, Mark, Emil, Francesco, Dušan supported me support by their precious recommendations during all my studies. I can not avoid mentioning all interesting discussions about new trends in Natural Language Processing methods with my friend Srikrishna Raamadhurai.

My happy moments would not be possible without my russian and foreign sun beams: Valerija, Anisa, Jana, Nora, Sayantan, Debdaus, Mathias and other friends. My first year in Finland was shared with Sayantan, Ehsan and Polina, to whom I am grateful for memorable experience inside and outside the university. Valerija supported me during my happy and difficult periods with her wise advices in spite of the long distance between us.

Last but not the least, everything would not be possible without my dear parents Rais and Roza, who raised me with love and strictness, and then let me move to Finland.

Espoo, October 15, 2014

Luiza Sayfullina

Contents

Chapter 1

Introduction

The analysis of public opinion about the trends in the social life, new types of products, marketing campaigns caught the attention both from academia and business side. In recent years, there has been much research done in the area of Sentiment Analysis as a strong marketing tool for analyzing client satisfaction about different products [32, 36, 39]. One of the first papers, devoted to Sentiment Analysis was the paper of Turney "Thumbs Up or Thumbs Down?" [39] in 2002, where he extracted opinions from movie and automobile reviews, finding semantic orientation. Turney has shown, that it is a strong marketing tool for better understanding the preferences of thousands people just from the publicly available social media. Twitter with more than 500 million of users, Facebook, information publicly available in diverse blogs are the sources of information for research in this area. Opinions about the products give the direction for good strategies. "What part of the new iPhone 5 was specifically successful and what should be changed in the next model"? Aspect-based Sentiment Analysis models are able to answer these questions, where opinion is found about specific items or people. Moreover, prediction tasks can be handled by mining of recent opinions. In other words, user preferences and opinion can be used for election prediction, for movie Oscar nomination, *etc.*, based on the already expressed opinions from public sources. In 2012, before the presidential elections for Republican Party in USA, research community used Twitter data in order to predict the election results[36] in each state by analyzing daily tweets. The constructed model was able to perform relatively good comparing with the official poll and predict similar voting results only for some candidates.

In the following subsection the classification of Sentiment Analysis tasks is given along with the approaches to solve them.

1.1 Sentiment Analysis Tasks Classification and State of the Art Methods

Most of the great classical philosophers - Plato, Aristotle, Spinoza, Descartes, Hobbes conceived emotion as responses to certain sorts of events of concern to a subject, triggering bodily changes and typically motivating characteristic behavior[1]. Emotion gives us internal sensations that we feel. Endless happiness that person can feel after a big dream comes true sadness after failing to pass the driving license exam, and other types of emotions, we feel constantly. Emotion should not be mistaken for the notion of sentiment. Sentiment can be a result of the specific emotion. Sentiment is a thought, view, or attitude based mainly on emotion. Often we feel sentiment towards some phenomena. "How do we feel when we buy our new phone, what is the attitude towards new candidate for the post of president?". All these questions are about our sentiment.

We define Sentiment Analysis as a general task of recognizing the emotions from the text. It has a broad meaning, including fine-grained analysis of the emotions from the range of possible choices, like anger, sadness, excitement, *etc.*, or simply detecting positive or negative sentiment.

To begin with, we give a short overview of the tasks connected with Sentiment Analysis. Opinion Mining is a part of Sentiment Analysis that deals with extracting opinions about different phenomena. Opinions fall into two categories: subjective and objective. Subjectivity detection seeks to identify whether the given text expresses opinions (subjective) or reports facts (objective) [18]. Objective sentences are regarded as neutral in Sentiment Classification tasks and do not provide any opinion. Researchers are interested in analyzing subjective sentences to find attitude towards the object. That is why the first step to begin with is usually finding text with subjective information, in other words, subjectivity detection problem [27] needs to be solved. While the solving this task, potentially subjective terms are extracted. Once this step is accomplished, one of the following tasks can be chosen: Polarity Classification, Emotion Identification and Sentiment Intensity Classification. We give an overview of these three tasks and tell about state-of-the-art methods proposed recently to solve them.

1.1.1 Polarity Classification

Polarity Classification is a task of predicting one of two opposite sentiments about an issue or phenomenon. "like" and "dislike" buttons, "thumbs up

[1] http://plato.stanford.edu/entries/emotion/#2

and down" are typical examples of opposite sentiment. The main condition is that sentiments express opposite meanings, like excited and bored, joy and anger, *etc.* In my thesis I mainly focus on this task, as a binary classification problem.

For supervised Sentiment Analysis there are two common approaches: Machine Learning and dictionary-based approach. Unsupervised approach was presented in several papers [40], but it is out of the scope of my thesis. Machine Learning approach is based on solving the classification problem with some set of features. In "SemiEval - 2013 Sentiment Analysis in Twitter" competition several teams classified tweets into three main sentiment categories: positive, negative and neutral [23]. The features used by teams included word-related features (1 - 3 grams, stems), punctuation (exclamation, question marks), syntactic (part-of-speech tags), as well as Twitter specific features (hashtags, urls, emoticons, slang words, repetitions). The choice of the classifier usually fell on the Support Vector Machines classifier [9] and Naïve Bayes classifier [24].

Machine Learning approach for solving text classification problems requires careful selection of the features. Irrelevant features create noise for the classifier, making the problem size large and decreasing the performance [12]. That is why it is preferable to make a subset of relevant features before applying the classifier. Information Gain [8], Chi-Squared Distribution [8], Mutual Information [19] are commonly used classical approaches for feature ranking. At the same time, it is not always clear how many features should be selected and how to avoid overfitting. Overfitting may happen when the accuracy on the test set significantly drops because the goodness of selected features was checked on the validation set. Indeed for validation set many features can be irrelevant, that is not the case with a test set. That is why a tight restriction on the number of features can lead to overfitting. Except that, Feature Selection can help to solve the problem of dataset sparsity.

Sparsity is one of the biggest problem of the constructed feature space [33]. The problem of sparsity originates from text representation. Imagine that each document is represented with a term-frequency vector, where each coordinate is responsible for the presence of the word in the document. The length of the vector is the size of the used vocabulary. Naturally, most of the values in each vector will be zero due to the large vocabulary and a relatively short document. Huge vocabulary can originate from the sources with informal text, spelling mistakes and lots of abbreviations. In this case, words follow Zipf's law [16]. Imagine each word w_i has a rank r_i, where ranking is made by the frequency of the word in the language and $r_j = 1$ corresponds to the most frequent word w_j. Then according to Zipf's law the frequency is inversely proportional $O(\frac{1}{r_i})$ to rank of the word w_i. Practically,

that means that the frequency of every next word in the rank decreases significantly.

In my thesis I address the challenge of reducing sparsity in textual data. Usually sparsity can be reduced to some extent by Feature Selection and Dimensionality Reduction. Traditional Dimensionality Reduction methods in Machine Learning, like PCA [37] or MDS [38] do not take into account the linguistic nature of the features. Moreover on high-dimensional sparse data with term-frequency representation applying PCA or MDS leads to information loss. Moreover, it makes difficult feature interpretation for some classifiers, like ELM [3]. The weakness of Feature Selection is that information from rejected variables is lost. That is why only features, carrying low amount of information can be omitted. On the other hand we can try to take into account semantic similarity of the words. For instance, we can try to merge such adjectives as "amazing", "nice", "wonderful" into one feature, that will represent adjective with positive feedback about something. The value for this feature in case of term-frequency representation sum up the frequencies of all three words, making the data less sparse. We present this approach later in the description of Word Clustering. For now we present some of the proposed methods to alleviate the sparsity.

In [33] the question how to alleviate data sparsity for Sentiment Analysis was highlighted. Their approach included extending feature space with sentiment-topic clusters [17], obtained through Joint Sentiment Topic (JST) model. JST model is an extension of Latent Dirichlet Allocation (LDA) [5], that groups words by topic and sentiment at the same time. LDA is a generative model, where document as a mixture of topics can be generated based on the topic-document distribution, and word-topic distribution. JST incorporates sentiment layer, thus having four layers totally. Sentiment labels are associated with documents, under which topics are associated with sentiment labels and words are associated with both sentiment labels and topics. Each cluster, obtained by JST model extends original feature space. In this approach, words within the same cluster are grouped both semantically and by sentiment, providing additional information in a very condensed way. The example of the sentiment-topic clusters mentioned in the paper is presented on Figure 1.1. In [33] 86.3% of accuracy was achieved in Polarity Classification task for Stanford Twitter Sentiment Dataset[2], comparing to 81% using only unigrams. The number of topics should be set in advance for the JST model. However, it is difficult to guess in advance what should be a proper number of topics. Hence, the quality of clustering can be checked only by observing the clustering result and analyzing how words are grouped

[2]http://help.sentiment140.com/for-students/

	Topic 1	Topic 2	Topic 3
Positive	dream, sweat, train	song, listen, love	eat, food, coffee
Negative	feel, toy, hate	rain, bike, car, stop	exam, school, weak

Figure 1.1: JST sentiment-topics clusters. Words are grouped both by topic and sentiment.

well together.

1.1.2 Emotion Identification

In Emotion Identification specific emotions from a predefined category are detected [7]. Group of emotions, proposed by Magna Arnold [29], including anger, aversion, despair, fear, *etc.* was utilized by some researchers, for example, in SentiSense tool. SentiSense is a concept-based affective lexicon, that attaches 14 emotional categories, represented in Table 2.4, to concepts. Using concepts, carrying particular meaning helps to avoid disambiguation. Indeed, one word can have several meanings and assigning the emotion to a specific concept is a more precise way. To illustrate, the word "miss" can have both positive meaning: "I miss you so much" and negative: "I missed the bus". In this example, two concepts of the word "miss" are presented. Concepts in SentiSense dictionary are taken from WordNet. WordNet [22] developed in Princeton is a lexical network, in which similarities between the concepts can be found. SentiSense dictionary was employed, because the vocabulary contains only emotional words, that are quite significant for classification. Each emotional category is regarded as a separate feature.

1.1.3 Sentiment Intensity Classification

In *Sentiment Intensity Classification* both the degree of the positive and negative sentiment is calculated for the same text. Indeed, even one sentence can contain at the same time both positive and negative sentiments about different parts of the phenomena. For example, one could state: "The camera of my new phone is amazing, but the battery life could be better". Here two opinions about different parts of the phone are expressed. Fine-grained analysis of the sentiment has been successfully done in the SentiStrength [14] tool. This tool was developed based on the dictionary or lexicon-based approach.

In this approach various lexical resources assign a sentiment score in different scales (+1,-1), (+5,-5) to the words or some categories. The most popular among them are SentiSense [10], SentiWordNet [2], AFINN [26] that

are made manually for English vocabulary. AFINN dictionary contains 2477 words, where each word has an integer score ranging between -5 (very negative) to 5 (very positive).

The implementation of the Sentiment Strength Detection Algorithm gave me a lot of insights and ideas while constructing features for the classifier. Some of the rules and details of the algorithm are listed here:

While constructing the features for solving Polarity Classification, I have used mentioned lists of words (vocabulary, negating, boosting list, *etc.*), provided by the tool and available for downloading. Not surprisingly, but SentiStrength algorithm for MySpace comments outperformed with 96.9% of accuracy many popular models, including Naïve Bayes (91.4%), SVM and the others. At the same time, the classification of positive and negative sentiment on the scale from 1 to 5 could be implemented even without labels using the existing Sentiment Strength Word list. The strength of the method is taking into account a lot of hidden rules, that are well-understood by humans, but not so well by machines. Secondly, fine-grained analysis of the sentiment for supervised algorithms requires a lot of labeled samples. Sentiment Strength can be potentially made in unsupervised manner. On the other hand, weak point is the dependency on the language. In other words, for each language new sentiment dictionary should be introduced. In addition, new rules should be created, taking into account language. In [1] Entropy Weighted Genetic Algorithm was proposed for Sentiment Analysis for several languages.

Due to the popularity of this topic, many challenges have been organized in order to attract more people and new ideas, including SemEval 2013 Sentiment Analysis in Twitter[3], Concept-Level Sentiment Analysis Challenge[4] and others. In the business section, several tools have been developed to help the companies to analyze customer opinions. We can name SenticNet[5], Luminoso[6], Factiva[7].

1.2 Problem Formulation and Structure of the Thesis

The scope of this thesis lies in the area of data sparsity analysis in Polarity Classification. Many researchers proposing state-of-art methods in Senti-

[3]http://clic2.cimec.unitn.it/starsem2013/

[4]http://2014.eswc-conferences.org/important-dates/call-SemSA

[5]www.sentic.net

[6]luminoso.com

[7]dowjones.com/factiva

ment Analysis are trying to achieve high performance by using a bunch of large amount of features, including n-grams. They use classifiers like Naïve Bayes which can handle high-dimensional data. At the same time, many of these features can be irrelevant and dublicated, decreasing the performance. Sometimes there is not enough data for making the classifier understand the relation between words. I want to investigate how can we reduce the sparsity in an efficient way: reducing the dimensionality but keeping as much as possible of the initial amount of information. There is a lot of information hidden in the linguistic nature of the features that gives room for introducing new approaches, not used in traditional Machine Learning, being able to combine similar features and choose the most relevant ones.

In the thesis I address not only the problem of sparsity itself, but also all the steps required to decrease it from setting the vocabulary, choosing data representation to Feature Selection/Reduction methods and their underlying strategies. The following aspects are observed and question are asked in the thesis:

1. How to construct the vocabulary at the early stage, which groups of words to include. How to use predefined sentiment dictionaries?

2. How is it better to represent the data? In term-frequency matrix or with special coefficients?

3. To explore which methods in Machine Learning and NLP are suitable for Feature Selection/Ranking. Based on the experiments, I have chosen linguistic approach: feature ranking based on unsupervised **tf_idf** score and Machine Learning approach LARS for deeper analysis.

4. To modify existing method to achieve better performance. Supervised version of **tf_idf** score was introduced. I compared the sparsity on chosen feature ranking approaches: LARS, used often with ELM, and both supervised and unsupervised versions of **tf_idf**.

5. To introduce an algorithm for merging semantically close words, being able to decrease both vocabulary and the sparsity significantly.

6. To see how Extreme Learning Machine is able to handle Text Classification tasks and compare its performance on different feature ranking approaches. I want to investigate how the sparsity of the data affect the accuracy and how many dimensions compared with the original feature size give the highest performance.

The sparsity problem is investigated on Polarity Classification task. Two completely different datasets are used for the experiments. One of them is taken from SemEval 2013 competition and contains informal text from Twitter. The second one is chosen from Kaggle website "KDD Predicting Excitement at DonorsChoose.org". My research on Sentiment Analysis has started from SemEval dataset. Later on some conclusions were made and prediction of the excitement involved modified approaches due to new insights. In addition, the second dataset is quite different, being written by teachers using formal language. The thesis after Introduction part is divided in two main chapters: Polarity Classification using Twitter dataset and Excitement Prediction using KDD dataset. First of all, I decided to include both datasets as I started my research on Twitter data, that has a lot of challenges to overcome. Second dataset was chosen later from Kaggle competition and contains long formal text within the specific topic.

The main classifier in the thesis is Extreme Learning Machine [3] that is able to handle high-dimensional data with low computational complexity. It has a reasonable accuracy compared with SVM. Despite the fact that most of the papers in Sentiment Analysis exploit SVM [9], we show that Extreme Learning Machine (ELM) are even more suitable for this task. ELM, thanks to low complexity, can be exploited for validating the proper number of ranked features to use in a reasonable time, resulting in higher accuracy.

The structure of the thesis is the following. In Chapter 2, on SemEval Twitter dataset I illustrate all the steps needed to be done before applying classifier, including preprocessing, vocabulary construction and data representation. I show how sentiment dictionaries can be used during the vocabulary construction. Moreover, I show how to apply Machine Learning FS approaches, like LARS, Information Gain, *etc.*, after vocabulary construction to reduce the sparsity. Supervised and unsupervised version of **tf_idf** is presented as a easy and powerful way of feature ranking. In the end, a short description and motivation for choosing Extreme Learning Machine as a classifier is given and experimental results are shown. Chapter 3 is dedicated to Excitement Prediction of essay data taken from KDD Challenge. Textual data of this dataset is compared to SemEval dataset and new more applicable ways of reducing the sparsity are provided, including Word Clustering by SubSequence Matching and WordNet dictionary. After we make an analysis of observed feature ranking, their strategy and how they affect the sparsity and performance. All the concluding remarks are provided in the last Conclusions chapter.

Chapter 2

Polarity Detection on Twitter Dataset

My interest in Sentiment Analysis started from trying to predict positive and negative sentiments from tweets. It was a challenging task, because the tweets are written in informal style. Moreover, in order to apply supervised method, good labeled dataset is needed. Choosing the right kind of dataset is quite an important step before making the experiments. As we have already discussed, labels acquired simply by emoticons are not reliable. In my first experiments I used Standford Twitter Sentiment Corpus, labeled automatically from emoticons. However, by manual inspecting of the tweets, I understood that the data is unreliable and labels can be controversial to meaning. In [14] the authors have made a good overview of the existing datasets for Sentiment Analysis and proposed their own dataset STS-Gold. STS-Gold dataset is based on the Standford Twitter Sentiment Corpus, that was annotated automatically by emoticons. However, in STS-Gold not only the tweets, but also 58 popular entities within each tweet were annotated with sentiment label. 2000 tweets were selected, containing at least one word from chosen entities plus 200 random tweets. The sentiment towards each entity was manually labeled. That solved the problem of a tweet with mixed sentiment, because both sentiments are labeled in a tweet if there are two sentiments towards different entities. Moreover, this dataset makes possible Polarity Classification both on the tweet and phrase levels. Although the dataset is manually labeled, the amount of samples is quite low for training a classifier. That is why, I have chosen larger SemEval dataset with also manually labeled tweets. The dataset was made as a part of the Semantic Evaluation (SemEval) workshop in 2013. Workshop included 13 different tasks targeting at evaluating computational semantic systems, including Temporal Annotation, Sentiment Analysis, Spatial Role Labeling, Noun Compounds, Phrasal Semantics, Textual Similarity, *etc.* One of the task was Sentiment Prediction of the tweet. SemEval dataset contains comparatively high num-

Sentiment Class	Tweet
positive	Gas by my house hit $3.39!!!! I'm going to Chapel Hill on Sat. :)
negative	Homegrown talent missing on Signing Day: Throughout most of the day on Wednesday, the video scoreboard ...#Raleigh
positive	One of my best 8th graders Kory was excited after his touchdown today!! He did the victor cruz!!lol pic.twitter.comtqORFrXB
negative	It's midnight on the east coast which means its @nickjonas birthday! HAPPY 20th BIRTHDAY NICK J!!!!!!!!!! ¡333

Table 2.1: Example of tweets taken randomly from SemEval Dataset. Tweets are characterized by the presence of slang, emoticons, shortened words, links, hashtags, *etc.*

	avg. # of words	pos. samples	neg. samples	voc. size
Tweet train	25.4	3045	1209	22012
Tweet test	25.4	1572	601	11376

Table 2.2: Description of SemEval dataset. The dataset is unbalanced, the number of samples wit positive sentiment is much higher than for negative sentiment. The vocabulary size is large, however average length of tweet is very short, 25 words.

ber of training and testing hand-annotated tweets, having multiple topics. One of the challenges of the dataset is its sparsity. In [14] the sparsity of several most popular datasets for sentiment analysis was compared. SemEval dataset had the largest sparsity among those datasets. The level of sparsity affects the performance of the classifier. However, as one of the goals of my thesis was the alleviating the sparsity, this corpus was the right choice.

From Table 2.2 we see that the vocabulary size or original dimensionality of the dataset is around 10 times higher than the number of samples. In the next section, we explain all the main steps in preprocessing of the tweets, including lemmatization and mistake correction.

2.1 Preprocessing

2.1.1 Lemmatization and Stemming

Twitter messages having informal style should be carefully preprocessed in the first step. Preprocessing of the textual data means filtering the text from the unnecessary tokens or words and putting the words and other symbols

to a unified form. This step is crucial for feature selection, because we need to keep only the relevant information in the consistent form. Consistent form means unified word representation and shrinking several forms of one word into one feature. For instance, words "funded", "funds", "funding" correspond to basic form "fund". Stemming or lemmatization [8] are used in Language Processing to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. In the context of bag-of-words model, that leads to feature dimensionality reduction.

Lemmatization is the process which creates the set of lemmas of a lexical database[1]. Lemmas are dictionary entries. Thus during the lemmatization, inflected words are corresponded to their dictionary entry form. Lemmatization removes only inflectional endings, resulting in keeping the same meaning of the word, but neglecting the form of the word. Let's say different forms of the verb "work": "working","worked","works" will end up as one lemma "work". The advantage is that the semantic meaning of the word is kept, that makes lemmatization reliable. Though the shrinking of the vocabulary is smaller than with stemming.

On the other hand, stemming aims at obtaining the stem of a word, that is, its morphological root. It clears the affixes that carry grammatical or lexical information about the word [34]. During the stemming the meaning of the word can be lost. The reason is because affix can convey meaning and some of the words have the same root or beginning. For, example words: "station", "static" and "stated" differ semantically, but having the same stem "stat". For some group of words, stemming can provide rough results by avoiding grammatical information.

Although stemming algorithms are generally rough, recent Lancaster module, provided by NLTK tool[2] works well and often keeps affix of the word. I have checked the stemming on the words: "hand", "handy", "handout" and "handle" and the stems of all 4 words were different: "hand", "handy", "handout" and "handl" respectively. Still, the problem with the same root existing in stemming was not handled by this algorithm.

To be more precise, I decided to implement lemmatization on the first step of preprocessing to avoid potential disambiguation and to merge semantically similar words later.

[1] http://www.christianlehmann.eu/

[2] http://www.nltk.org/api/nltk.stem.html

2.1.2 Preprocessing algorithm

Taking into account that tweets contain some specific punctuation, many links, urls and retweets, I applied the following steps to clean the text and extract words and punctuation:

1. Deleting urls and hashtags
2. Tokenizing the words though NLTK tokenizer[3]
3. Filtering the words
   - Removing stop words
   - Removing repeated groups of symbols

Tokenizer is a tool dividing a text into a sequence of tokens, which roughly correspond to words and punctuation [8]. After extracting units produced by tokenization, the filtering step follows. Removing stop words, the ones that are short and do not convey the meaning, decreases vocabulary size. In order to check if the word is in the stop-word list in English, NLTK.corpus.stopwords module in python was used.

Repetitions used in informal language should be also reduced to the basic form of the word. Imagine, we have several versions of the word "hey": "hey", "heyiheeeyyy","heyhey". Deleting simple vowel repetitions does not solve all the cases, therefore I have applied recursive procedure in order to remove all repeated parts of the word. The pseudocode of the algorithm is presented in the Algorithm 1. In *If* condition *Punctuation* are strings containing at least one punctuation symbol. We do not want to analyze the string if it is informal and has some punctuation in it, for instance, "haha!!".

Algorithm 1 Recursive repetition elimination

```
1: procedure RECURSIVE REPETITION ELIMINATION ALGORITHM(S)  ▷ The string to remove repetitions from
2:     if S is spelled correctly and S ∉ Punctuation then  ▷
3:         while Repetition in S are found do
4:             Delete the repetition
5:         end while
6:     end if
7: end procedure
```

[3]http://www.nltk.org/api/nltk.tokenize.html

After the vocabulary and punctuation are filtered and tokenized, the following step is to construct the list of the words and other lexical units used for classification. We are going to describe this step in the following section.

2.2 Feature Engineering

In order to apply a classifier, every sample should be represented through some predefined set of features or variables. In Text Classification one of the popular model is Bag-of-Words model [8], where exact ordering of the terms in a document is ignored and each document is represented by the number of occurrences of each term in the vocabulary. In this case, the frequency of the word serves as a feature.

There are two ways of constructing the vocabulary:

1. Only using the training dataset and perform feature selection

2. Using training dataset and words from predefined sentiment dictionary

Overfitting can occur in feature selection using the training dataset. In other words, the classifier can perform much worse on the new dataset [4]. Words from sentiment dictionaries can serve as a good base for vocabulary, not depending on the dataset, as they provide words expressing emotions.

Thus we construct the set of features, based on the sentiment dictionaries and training dataset. Using training dataset helps us to find important domain-specific features, and words from ready sentiment dictionaries serve as base features that do not depend on the training set.

While constructing feature set I used SentiStrength Software Tool[4] dictionary items. Words in the dictionary are grammatically and semantically grouped in the following categories: BoosterWordList [”very”, ”too”, ”bit], EmoticonLookupTable, IdiomLookupTable, NegatingWordList, QuestionWords, SlangLookup, EmotionLookupTable. In the main file EmotionLookupTable the words are given positive or negative scores from 2 to 5. In my approach these scores are not employed, instead the words themselves are used as features. EmoticonLookupTable listed emoticons, made from punctuation symbols and their sentiment scores from -1 to 1. From the listed word lists not all were taken, but only those, that improved the accuracy on the validation dataset.

Except SentiStrength dictionary I selected the features from SentiSense dictionary. As already has been described, in this dictionary words are

[4] `http://sentistrength.wlv.ac.uk/`

grouped according to 14 emotional categories: joy, sadness, love, hate, despair, hope, *etc.* These 14 emotional categories also were chosen as features, where the value for each feature represents the amount of words, falling into each category. Because the groups are classified according to the sentiment this gives us good sparsity reduction. The set of 10 opposite emotion pairs is presented in Table 2.4.

Moreover, I have extracted with AlchemyApi[5] tool the categories: celebrities and places. People quite often have some opinion about celebrities. Therefore for some celebrity at different periods of time opinions are more positive, because of the burst of popularity, amazing new album or more negative sentiment can be assigned. Celebrities and places are quite domain and dataset specific features.

SentiStrength and SentiSense dictionaries provided good ground of features. Frequent unigrams and bigrams were chosen from the training dataset. In general case, N-gram is a phrase, consisting of N words. Bigrams are able to capture the word order, however the amount of possible bigrams is large. That is why Feature Selection needs to be done. We describe later what Natural Language Processing and Machine Learning Feature Selection (FS) and Dimensionality Reduction (DR) were applied to alleviate the sparsity problem.

In Table 2.3 different groups of features for classification are presented. Unigrams has the largest size. The number of bigrams is small, because I selected only the ones, that has high Pointwise Mutual Information. In Table 2.5 some examples from these groups of words are presented. Only the groups of features, that contributed to the improving the accuracy on the validation dataset, were included. Among them are: Unigrams, Bigrams, Slang words, Negations, Emoticons and Syntax.

2.3 Data representation

In this section we give an overview of **tf_idf** weights used much in Information Retrieval [19]. In section 2.4.2 we describe how weights can also be used for Feature Selection, and present supervised version and unsupervised **tf_idf** scores. For now, we explain **tf_idf** and give an intuition for it. In Text Classification problems document indexing is the creation of numerical representation of the document. It consists of two steps:

1. Term Selection - finding the terms, important for Text Classification. In Machine Learning this term refers to Feature Selection.

[5]www.alchemyapi.com

Feature	**Number of items**	**Cumulative sum**
Bigrams	78	78
Topics	14	92
Slang words	90	182
Booster words	28	210
Negations	17	227
Prepositions	36	263
Emoticons	36	299
Syntax	87	386
Celebrities	91	477
Places	100	577
Links	18	595
Unigrams	2197	2792
Totally	**2792**	**2792**

Table 2.3: Features for sentiment detection

2. Term Weighting - assigning a value to each term in the document to represent the contribution of this term for distinguishing the document from the others.

Term frequency **tf** [19] is calculated by using the frequency of the term in the document **t** divided by the length of the document $|D|$:

$$\mathbf{tf}(term, document) = \frac{\mathbf{t}}{|D|}. \tag{2.1}$$

However, the frequency of the word does not always show its importance for classification, as there are many stop words which are commonly used in the documents but do not carry semantic meaning, like "and", "still", "you", *etc.* That is why **idf** weight was introduced [19]:

$$\mathbf{idf}(term) = \log(\frac{N}{\mathbf{df}}), \tag{2.2}$$

where **df** denotes the number of documents that contain the corresponding term and N is the total number of documents. In other words, it is the inverse document frequency calculated for each term. Document inverse frequency **idf** assigns more weight to rare terms.

The combination of the term and inverse document frequency led to **tf_idf** weight [19] defined as:

$$\mathbf{tf\_idf}(term, document) = \mathbf{tf}(term, document) \cdot \mathbf{idf}(term). \tag{2.3}$$

Negative class	Positive class
acceptance	refusal
anger	calmness
anticipation	surprise
aversion	desire
courage	cowardice
dejection	hope
despair	hope
disgust	like
fear	calmness
hate	love
joy	sadness

Table 2.4: SentiSense emotional categories

This weight is intuitive, as both **tf** and **idf** weights contribute to word importance in the document. At the same time, the ideal weight is achieved when there is a proper balance between the term frequency and the number of documents, where this term occurs. Rare terms have high **idf** but low **tf**. Stop words follow the opposite rule. Words having high **tf_idf** weight should occur only in set of specific documents, giving them some semantic meaning.

To conclude, **tf_idf** weight was constructed on two following assumptions:

1. The higher is the frequency of the term in the document, the more important it is.

2. Terms that occur in smaller amount of documents are more significant.

2.4 Dealing with Sparsity: Dimensionality Reduction methods for Textual Data

In the following section we describe, how to make the selection of important terms by Machine Learning methods, like LARS [11], Information Gain [35] and Mutual Information [8]. Moreover, we talk about alternative ways of Feature Selection/Ranking in Natural Language Processing using **tf_idf** score and its supervised version.

Features added	**Examples**
Unigram	desirable, protest, nursery, sleep, hating, hate, disability, abundance, sorry
Bigram	don't, tell me, my mom, so excited, found, out, so much, good morning, just found, did n't, your ticket
Topics	withhappiness - joy, teaching - like, magnet - anticipation, sufficient - anticipation, gloomily - sadness, gun - fear
Slang	aight, alol, b4, b4n, bak, cu, cya, cyo, fud
Booster	absolutely, completely, very, might, could
Negations	aren't, are not, couldn't, do not, wont, would not
Prepositions	against, anti, but, despite, down, during, except, inside
Emoticons and syntax	:), ;D, ((, xD , ..., !, ?
Celebrities, places, links	julia gillard, mick jagger, chinese paris, maryland, vegas, hawaii, instagr.am, bnds.in, adf.ly

Table 2.5: Vocabulary items used as features for classification. Features are grouped according to some criteria. Grouping of words is useful for Feature Selection that is done by including or removing specific group of words. This applies mostly to potentially not significant groups of words, such as prepositions, city names, *etc.*

2.4.1 Feature Ranking approaches

The performance of the classifier is closely connected with the sparsity of the dataset and number of dimensions. In practice, for example, in high-dimensional space distance between two samples in high-dimensional space tends to be approximately the same for different pairs of points. This is one of the effects of the curse of dimensionality [4].

In order to deal with the sparsity we should know the methods of measuring it for textual dataset. Let $\mathbf{v} = [v_1, v_2, ..., v_n]$ denote the vector with vocabulary size v_i for each tweet i, where n is the total number of tweets and $|V|$ is the size of the dataset vocabulary V. Sparsity degree can be calculated as the following:

$$S_d = 1 - \frac{\sum_{i=1}^{n} v_i}{n * |V|}. \tag{2.4}$$

The degree of sparsity $S_d \in [0, 1]$. The higher is the degree of sparsity of the dataset, the more potentially challenging is the task for the classifier. While decreasing the sparsity it is crucial to keep as much as possible the information of the original feature space. That means, there should be a balance between reducing the number of variables and keeping information. One of the ways how the sparsity can be reduced is merging words, having the same semantic meaning. In this case, the information is not lost. Another way of reducing the sparsity is getting rid of terms not contributing to classification, like stop words [31]. LARS [11] and Information Gain [35] are approaches that can be used for feature ranking and therefore can be used for reducing sparsity.

2.4.1.1 Least Angle Regression

Least Angle Regression (LARS)[11] is a Feature Selection method that can be used for feature ranking. Let $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}$ be the variables and $\mathbf{y}$ the output vector. We want to construct the model by adding subsequently the variables having the largest correlation with the residual $\mathbf{r}$, defined in the following steps.

1. Set all variable coefficients $\beta_1, \beta_2, ..., \beta_n = 0$, current residual $\mathbf{r} = \mathbf{y}$.
2. Find variable $\mathbf{x_j}$, $j = [1, n]$ that has the highest correlation with the residual.
3. Increase the coefficient β_j in the direction of covariance or correlation between $\mathbf{x_j}$ and current residual vector. While increasing β_j calculate the residual $\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}$, where $\hat{\mathbf{y}}$ is the estimate using all added variables and estimated β coefficients.

4. Stop, when some variable $\mathbf{x_k}$ has the same correlation with residual $\mathbf{r}$ as $\mathbf{x_j}$.

5. Move in the direction equiangular between all added variables or between current estimate and the last added variable. Go to step 2.

As a result we get the list of n ranked variables $\mathbf{x_j}$ with their coefficients β_j.

For feature ranking LARS method can be calculated on the training labeled data. LARS continuously chooses the variables according to the ranking. The earlier feature is selected, the higher ranking it has. One of the advantages of LARS, compared with using only correlation between the variable and the output, is taking into account the residual $\mathbf{r}$ while adding a new feature. This means that each new added variable should bring new information. The number of the top ranked features should be cross-validated on the training data and specific model, that is used for classification. LARS method was not only used with ELM, but also for Text Classification [15].

2.4.1.2 Information Gain

In [35] Anuj Sharma *et al.* have compared different Feature Selection methods for Text Classification: Information Gain, Document Frequency, Gain Ratio, Chi-Squared and Relief-f. Information Gain and Gain Ratio showed the best performance for Feature Selection in Polarity Classification.

Each feature as variable gives certain amount of information after we observe its value. The more variable's value is unexpected, the more information variable has. In order to measure the amount of information, we need to use its probability distribution $P(x)$, that expresses the probability of a specific value of the variable. The amount of information can be expressed as:

$$h(x) = -\log_2 P(x). \tag{2.5}$$

The entropy as the average amount of information for a probability distribution P is given by equation:

$$H(P) = -\sum_x P(x) \log_2 P(x). \tag{2.6}$$

Entropy is calculated as the mean or average value of the information received with respect to the distribution. For a random variable X with values in the finite set, $H(X)$ denotes the entropy of the distribution of X.

Information Gain is the measure of how much uncertainty about the variable we decrease after observing the value of the variable x. It is calculated as follows:

$$IG(c, x) = H(c) - H(c|x)\,, \tag{2.7}$$

where c is the class label variable. Feature conditional entropy, in other words, information after knowing the feature can be calculated in the following way:

$$H(c|x) = -\sum_{i=1}^{n} P(c = i|x) \log_2 P(c = i|x). \tag{2.8}$$

In Polarity Classification there are two class labels, representing the sentiment: positive and negative $c \in \{+1, -1\}$.

Information Gain technique selects the features having the highest difference in entropy after partitioning the instances according to this feature. One by one feature x giving the highest information gain $IG(c, x)$ is added. In Polarity Classification we select such set of words that makes us closer to guessing the sentiment of the tweet.

IG was specifically useful while selecting the set of frequent unigrams from training set vocabulary. Since the amount of unigrams is large, IG is able to reduce the size of the vocabulary to some extent. At the same time, the attempt to use IG on the whole set of features did not result in good performance.

2.4.1.3 Pointwise Mutual Information

Mutual Information [8] measures the information overlap between two variables x and y:

$$MI(x, y) = H(x) + H(y) - H(x, y). \tag{2.9}$$

Using the definition of the entropy, the equation can be presented though probability distributions:

$$MI(x, y) = \sum_{x,y} P(x, y) \log(\frac{P(x, y)}{P(x)P(y)}). \tag{2.10}$$

MI is the highest, when x and y are fully correlated and $H(x) = H(y) = H(x, y)$. In case, x and y are independent $P(x, y) = P(x)p(y)$, $MI(x, y) = 0$.

Pointwise Mutual Information (PMI) measures how much one word x tells us about the other y and it can be measured as $PMI(x, y) = \frac{P(y|x)}{P(y)} = \frac{P(x,y)}{P(x)P(y)}$. As we see from the formula PMI finds how much joint distribution and individual distributions deviate, assuming independence of the separate words. MI can be viewed as the average PMI.

NLTK tool has BigramCollocationFinder class, that allows to sort collocations according to their PMI score. Selection of the bigrams was made by PMI. After the preprocessing I extracted all possible bigrams, where both of the words did not contain any punctuation, and selected only 70 bigrams out of them due to general low PMI score. Some of the bigrams are presented in the Table 2.3.

2.4.2 Feature Ranking by Unsupervised and supervised tf_idf score

While talking about data representation in Section 2.3 we have already described **tf_idf** weights. Still, this weighting can be used not only for effective data representation, but at the same time as a supervised Feature Selection method. In other words, both the weights and labels are exploited in order to find the most significant features.

As we have already explained previously in Equation 2.3 the higher the $\mathbf{tf\_idf}_{ji}$ weight, the more important the term j is for distinguishing a document i. If we sum up the weights across the documents, we receive the score for the word importance across all documents, disregarding the labels:

$$\mathbf{uscore_j} = \sum_i \mathbf{tf\_idf}_{ji}. \tag{2.11}$$

This score is unsupervised, since class labels are not used to find the relevance of the word across the documents. I refer to it as unsupervised **tf_idf** score or simply **tf_idf** score.

Nevertheless, one term can be important for both classes and hence it does not help to distinguish between the documents. In order to find only the terms that distinguish one class from another, we propose a supervised version.

Let $\mathbf{y} = [y_1, ..., y_N]$ be the output vector, where $y_i \in \{+1, -1\}$, $i \in [1, N]$. $\mathbf{tf\_idf}_j$ denotes the row j from **tf_idf** matrix with scores for word j, $j \in [1, M]$.

Then the relevance score is calculated as follows:

$$\mathbf{sscore_j} = |\mathbf{y} \cdot \mathbf{tf\_idf}_j^T|. \tag{2.12}$$

In Binary Classification problem we need to find terms discriminating between two classes. Each element in $\mathbf{tf\_idf}_{ij}$ reflects how word i is important for j document. If we take the scalar product of the vector $\mathbf{tf\_idf}_j$ with class label vector $\mathbf{y}$, $y_i \in \{+1, -1\}$, then we sum up all weights for positive class and subtract the weights from negative class. As a result, if $\mathbf{sscore_j}$ score is

positive, then the word j is more significant for positive documents. However for us the sign of the $\mathbf{sscore_j}$ is not important as we are interested in finding the term discriminating either of the classes. Hence, the absolute score is taken into account. The lower the score, the less a word can distinguish between the documents from different classes. The most relevant features are chosen from the top scores.

I should note that the concept of supervised score was my novel approach, previously it has not been proposed in the literature. In addition, I decided to try modified version of **tf_idf**, proposed by Kao *et al.* [19]. They have modified this score and made it supervised. To describe formula, we need to define the following four scalars: A, B, C and D, that can be used for calculating Information Gain, Mutual Information and Chi-Square.

	Positive class	**Negative class**
t_k	A	B
$\overline{t_k}$	C	D

Table 2.6: Four fundamental scalar values A, B, C and D, used for calculating IG, MI. A and B means how many documents contain term t_k from positive and negative class respectively. C and D means how many documents do not contain term t_k from positive and negative class respectively.

To provide the inspiration for a new modified **tf_idf** score, we show the formulas for Information Gain and Mutual Information using four measures A, B, C and D:

$$\mathbf{IG} = -\frac{A+C}{N}\log(\frac{A+C}{N}) + \frac{A}{N}\log(\frac{A}{A+B}) + \frac{C}{N}\log(\frac{C}{C+D}), \tag{2.13}$$

$$\mathbf{MI} = \log(\frac{AN}{(A+B)(A+C)}). \tag{2.14}$$

One of the disadvantages is that **tf_idf** score is not taking into account the class label information. In [19] the authors analyze this score from local and global levels:

1. Local weight is **tf**, that shows the contribution of term to the specific document.

2. Global weight is **idf**, that shows the contribution of the term for distinguishing document in a global sense.

Authors proposed to use labels in **idf** score responsible for importance of a term on the global level. Global level means how important term is for discriminating particular class of documents. Two components were included in the modified score:

1. The ratio A/B, that shows to which degree the term occurs more in one class, than in another.
2. The ratio A/C, that shows for particular class how often the term appears there.

These two ratios are taken into account and incorporated into modified **tf_idf** score:

$$rel_{term} = \mathbf{tf} \cdot \log(1 + \frac{A}{B} \cdot \frac{A}{C}). \tag{2.15}$$

The resulting formula is more intuitive than Information Gain or Mutual Information if we look at the equations presented. Recall Equation 2.2 for **idf**, here it is replaced by the logarithm of the multiplied ratios. The logarithm in the formula is used in analogy with inverse document frequency formula, that also used logarithm. Logarithm helps to mitigate the value inside of it. In these ratios class label information is used. Though the quality of feature selection depends on the database, the authors in [19] used Reuters-21578 benchmarking collections, combining different engineering technical papers. In their experiments modified score outperformed traditional **tf_idf** score by 12% in F1-score.

2.5 Extreme Learning Machine Classifier

Extreme Learning Machine (ELM) [3] is a generalized Single Hidden Layer Feedforward Network (SLFN). In ELM the weights from input to the hidden layer need not to be tuned and back-propagation is not needed. Usually nodes in the hidden layer are initialized randomly and independently from the training data. At the same time output weights are solved using the Least Squares Method, that makes ELM extremely fast.

ELM has a universal approximation property stating that $\exists\, \epsilon > 0$, such that M neurons can be found to approximate N samples within the accuracy ϵ [3].

The model of ELM can be described as the following. We are given a set of N samples $(\mathbf{x_i}, y_i)$, $i \in [1, N]$, $\mathbf{x_i} \in \mathbb{R}^p$, $y_i \in \mathbb{R}$. The output of neurons in

the hidden layer can be calculated as:

$$\sum_{i=1}^{M} \boldsymbol{\beta_i} f(\mathbf{w_i^T x_j} + b_i),\ j \in [1, N]. \tag{2.16}$$

Here $\boldsymbol{\beta_i}$ is an output weight vector (between hidden layer and output), f is an activation function, $\mathbf{w_i}$ is an input weight vector and b_i is a bias. The equation for the output is considered as the following: (assuming the Neural Network approximates perfectly)

$$\sum_{i=1}^{M} \boldsymbol{\beta_i} f(\mathbf{w_i^T x_j} + b_i) = y_j,\ j \in [1, M]. \tag{2.17}$$

If we denote $h_{i,j} = f(\mathbf{w_j x_i} + b_j)$, then we get the following equation:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{y}. \tag{2.18}$$

Thus the output weights can be calculated by $\boldsymbol{\beta} = \mathbf{H}^{+}\mathbf{y}$, where $\mathbf{H}^{+}$ is the Moore-Penrose pseudo-inverse matrix of $\mathbf{H}$.

The advantage of using this classifier is that we only need to choose an activation function f and adjust the number of neurons in the hidden layer. Another advantage is that with the random initialization and with correctly chosen hidden-layer output matrix the target values can be approximated to any chosen accuracy ϵ. ELM might suffer from irrelevant variables, therefore feature selection should be applied in this model [21].

2.6 Optimally-pruned Extreme Learning Machine

Optimally-pruned Extreme Learning Machine(OP-ELM) [20] was developed in order to solve the cases, when ELM performed poorly with the presence of noise or irrelevant variables. The idea of the OP-ELM is to incorporate feature selection step in ELM. The algorithm consists of the three main steps:

1. Building an ELM with large number of neurons
2. Using LARS for ranking the neurons
3. Applying Leave-One-Out (LOO) validation for selection the number of neurons in the hidden layer

The advantage of the method is that during the first step calculating LOO error takes linear time, estimation of it has fast closed form formula.
It has been shown by Yoan Miche [20], that OP-ELM gives more stable results than ELM, that is more randomized. The complexity is higher than ELM, but at the same time computational complexity is several magnitudes lower than SVMs, for example.

2.7 Experimental results

Thus far, we have described the data representation, the methods, used for feature ranking and ELM classifier. Here I sum up, how the described approaches were used together and show some interesting experimental results.

First of all, for data representation, I have chosen term counts, not **tf_idf** weights. The reason is that ELM, as Neural Network is trained better with raw counts, instead of scores. Vocabulary was constructed using training set and sentiment dictionary. From training set I selected top unigrams by Information Gain, bigrams by Pointwise Mutual Information. SentiStrength sentiment dictionary provided me the list of slang words, negations, emoticons and syntax. Other groups of words, presented in Table 2.3, were discarded by ELM validation. If some specific group of terms decreased the performance on the validation set, it was discarded. ELM classifier due to its fast performance was a very good choice for making fast validation.

I set 70 neurons in the hidden layer by validation and hyperbolic tangent as an activation function in ELM. Training set is 2000 samples, validation is 445 and 815 samples are in the test set. The size of the vocabulary is 1217.

On Figure 2.1 I compare the accuracy between supervised and unsupervised **tf_idf** feature ranking on ELM. Supervised Feature Ranking **tf_idf** outperforms almost on the whole range of observed features. Supervised approach seems to work well in practice and later on the KDD dataset we prove this statement again.

Sparsity plot for Twitter data, consisting of unigrams, bigrams, syntax, emoticons, slang and negations, is presented on the Figure 2.2. We see how drastically sparsity was decreased with both **tf_idf** approaches, comparing to the original sparsity, equal to 0.9928.

We have presented top selected words, chosen by unsupervised **tf_tdf** approach on the Figure 2.3. Top words include the words, directly having emotional coloring, like "love", "anticipation", "hope", "joy", "fear", "disgust", as well as frequent words, e.g. "today", "tomorrow" and "tonight". On Figure 2.4 we made the same plot, however for supervised **tf_tdf**. Words shown are very similar, on the other hand we do not show all feature ranking,

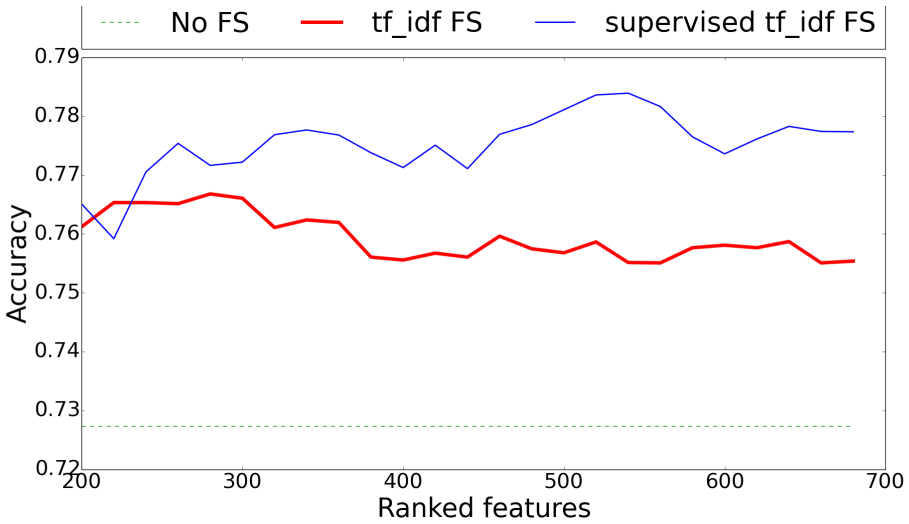

Figure 2.1: The accuracy of the ELM on the ranked by **tf_idf** features. In this example supervised selection outperforms on different number of features. The experiment is done on the test set. The highest accuracy is achieved for less than 580 features, that is less than a half from initial feature size.

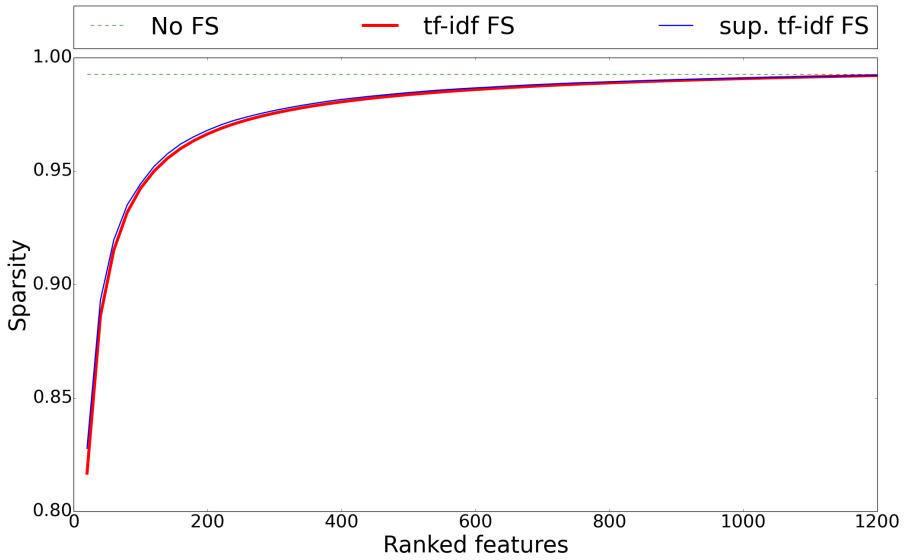

Figure 2.2: Sparsity plot on Twitter dataset. Both versions of **tf_idf** provide quite high level of sparsity, comparing to 0.9928 sparsity level of the whole dataset.

that has differences. This can be proven by the higher accuracy results on the same number of ranked features between supervised and unsupervised version.

Figure 2.3: Top selected words by **tf_tdf** approach in Twitter dataset.

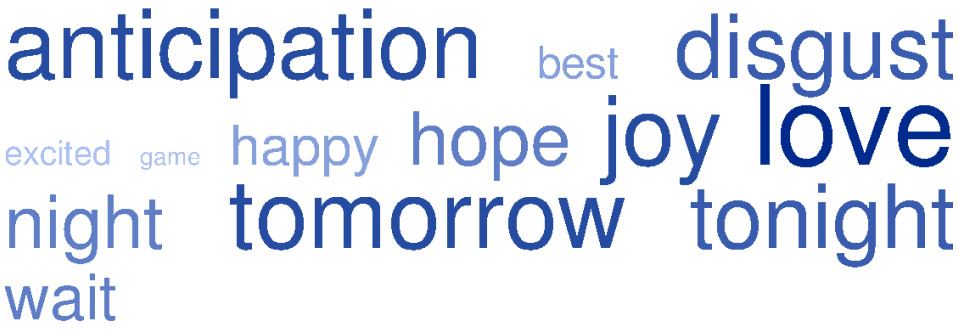

Figure 2.4: Top selected words by supervised **tf_tdf** approach in Twitter dataset.

Although we have presented LARS approach for Feature Ranking in practice for this dataset the method did not perform well enough for presenting the results in the thesis. However, we make more comparison of LARS with **tf_tdf** approach in the next chapter. We make detailed analysis of how actually they select the features, how they manage to reduce the sparsity and how reducing the sparsity affects the performance. For now, we move to the next chapter, dedicated to other Polarity Classification problem.

Figure 2.5: Top selected words by both supervised and unsupervised **tf_tdf** approach in Twitter dataset. As we also see from the sparsity plot features are overlapping much. However the performance on ELM of the supervised approach is still higher.

Chapter 3

Excitement Prediction on KDD Project Excitement Challenge

The third chapter is a continuation of solving the Polarity Classification problem, however on a completely different dataset. Instead of the sentiment we try to predict if the project is exciting, meaning it has received money from donations and other criteria. I introduce WordNet Clustering, grouping semantically similar words and decreasing the sparsity. Mostly, this chapter is dedicated to sparsity analysis before and after applying several observed methods: LARS, **tf_idf** and WordNet Clustering. I show that best performance was achieved with a combination of **tf_idf** and WordClustering. In the end the experiments are done on the classifier less sensitive to sparsity problem, Naïve Bayes.

First of all, I make a short description of the dataset, used for the experiments in this chapter.

3.1 Dataset Description

The dataset is chosen from challenge by KDD Cup, presented in Kaggle[1] site.

DonorsChoose.org[2] is an online charity that makes possible to help students in need through school donations. Teachers from schools propose projects requesting material to enhance the education for their pupils. After a project reaches its funding goal, the materials are shipped to the school.

The 2014 KDD Cup asks participants to help DonorsChoose.org predict projects that are exceptionally exciting to the business, at the time of posting. While all projects on the site fulfill some kind of need, certain projects

[1] kaggle.com

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

have outstanding ideas. By identifying and recommending such projects it is possible to improve funding outcomes, improve user experience, and help more pupils receive study materials. Dataset provided by the challenge is

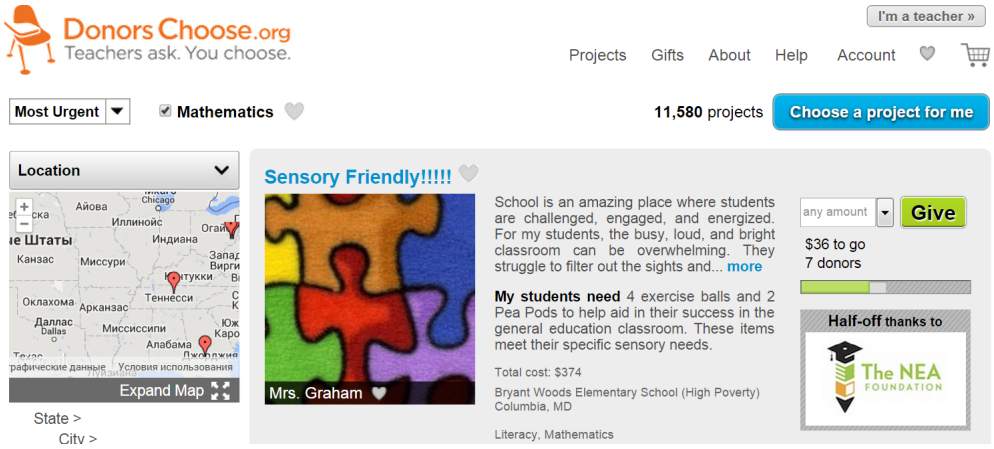

Figure 3.1: Example of a project at Donorschoose.org web site

quite different from Twitter dataset.

1. The description of the project is given in formal English, with negligible number of mistakes.

2. The vocabulary is diverse. The average length of the word is longer and the language is more formal. Partly, it can be explained by the absence of slang and abbreviations.

3. The length of the average project essay is larger, containing 100 - 200 words, after removing stop words. This makes the dataset less sparse. In Twitter the maximum length of the tweet is 80 symbols.

4. Dataset contains 2995 and 48457 essays of exciting and non-exciting projects respectively. This dataset has a larger number of labeled textual data. However, the proportion of the exciting projects to non-exciting is more unbalanced, comparing to Twitter dataset. In Twitter negative sentiment occurs in around 30% of cases.

3.1.1 Textual Data

One of the reasons why this dataset was chosen is the presence of both textual and numerical data. Textual data comprises the project description provided on the web page. Essay describes the current situation in a school

and explains how products or services are needed for enhancing education process. Quite often the reason for asking the money is the lack of necessary materials for study, like books for example. In the dataset textual data is separated into several parts for better analysis. In Table 3.1 the fields from essay are presented. Next I present one of the essay examples[3] "Music around the world" for better understanding.

Field name	**Description**
project id	unique project identifier
title	title of the project
short description	description of the project
need statement	need statement of a project
essay	complete project essay

Table 3.1: Fields from essay table. Need statement has a textual description of the requested things, how they will be helpful for education. Short description is the short version of essay. Essay contains the full description of the project.

[3]http://www.donorschoose.org/project/music-around-the-world/

Music around the world
My students A typical day in my classroom can describe as full of energy, filled with eager and ready to learn students. These students have never been able to experience a real musical classroom because of the resources that they had. As a third year teacher, I am trying to update our materials and resources.
We are located in South Carolina. These students are full of energy with a determination to learn. The school does not have a great reputation with music, and as a third year teacher, I am trying to change that. I feel that a main reason their reputation has not been positive is because of the limited and out-updated resources and materials that they have. They cannot be successful with the current state of the classroom...
My project The egg shakers will provide an opportunity for all my students to focus on rhythmic accuracy and provide a sense of tempo. The other items including the djembes, pipe instrument, and rhythm kit aids in the drumming circle as students are not able to have the current instrument support in my classroom. This overall project is key to overall success of the students. Try to remember your elementary music classroom and the amount of materials and resources you had. While not fully completed, this small contribution will help us in our quest for success...
If you, the donors, show our students that we care and appreciate them, they will develop a more sustainable work ethic that they can maintain in this world. These materials will open a brand new door to students showing off their creativity, which is something, that is lacking with this school and our students. I want to show students that they do have options in the way they can be creative and motivate. My students need these various items such as egg shakers, rope-tuned djembe, and a bamboo rhythm kit as part of the World Music Unit I am starting in the spring.

3.1.2 Project Data

Project data comprises several tables: resources, donations, projects. Donation table, where the details of donation transactions are described, was not used in my predictions. As one project has several donations, some model needed to be built in order to transfer donation data about the project into the project information. It was not in the scope of the thesis.

At the same time, resources table contained information about the required items, their price, quantity, *etc.* I regard this information as quite important for classification, as items requested by the teachers affect the decision for donation.

Project description table was merged with resources table by project id

and significant fields manually were selected at the first stage.

The amount of donations is not the only criteria for the project to be exciting. There are 5.9% exciting projects and 30.5% are fully funded. Except being fully donated the project has to receive the money from at least one teacher and to have higher than average number of donations, *etc.* Project Dataset as well as Essay (textual) Dataset is also unbalanced. However, taking into account that project dataset comprises 36710 exciting projects out of 600000 projects, it is large enough for making own balanced training, validation and test set.

The dataset was constructed by taking 5890 projects containing both essay and project description data. Data is balanced, meaning that it consists of the same number of exciting and non-exciting projects. 4000 samples belong to training set, 890 to validation and 1000 to test set. Essay Data is classified by ELM with hyperbolic tangent, 70 neurons in the hidden layer and random initialization of the weights between the input and hidden layer. The number of neurons in the hidden layer was taken from multiple experiments on the validation data. This dataset was used in all of the experiments, that I performed through the chapter.

The full list of project features is listed at the challenge website [4]. The information includes school information (location, availability of some resources), poverty level, grade level, the subject area, information about the teacher and the resources, for which they ask money.

3.2 Classification using Essay Data

In this subsection we make the analysis of how Feature Ranking affects sparsity and how sparsity affects the performance of ELM.

3.2.1 Comparison with Twitter Dataset

First of all, I make the comparison with the Twitter dataset in order to understand better vocabulary distribution before applying FS.

Let us define the vector $\mathbf{df} = [df_1, df_2, ..., df_M]$, where M is the vocabulary size and df_i denotes the number of documents containing word w_i. Words, having the same value df are equally frequent in the dataset. In order to understand better the sparsity of the dataset, the histogram of **df** vector is plotted. Each bin groups together the words with the close values of df.

At first I present the histograms of initial Twitter and KDD dataset in Figure 3.2 and 3.3 before FS method. In KDD dataset almost half of

[4] kaggle.com

the vocabulary words occur in more than 10 documents. In comparison, in Twitter dataset negligible number of terms occur in more than 10 documents. That means, that the features in a new unseen dataset can not be covered well enough in the training set. Sparsity is also connected with the text, coming from diverse topics and people in Twitter. It is clear, that people having different interests and hobbies will share diverse information publicly and frequently used words can differ significantly between various categories of people. At the same time, web site oriented on people making the donations for educational purpose has a common topic. That is why, the vocabulary words are more likely to repeat in several different essays.

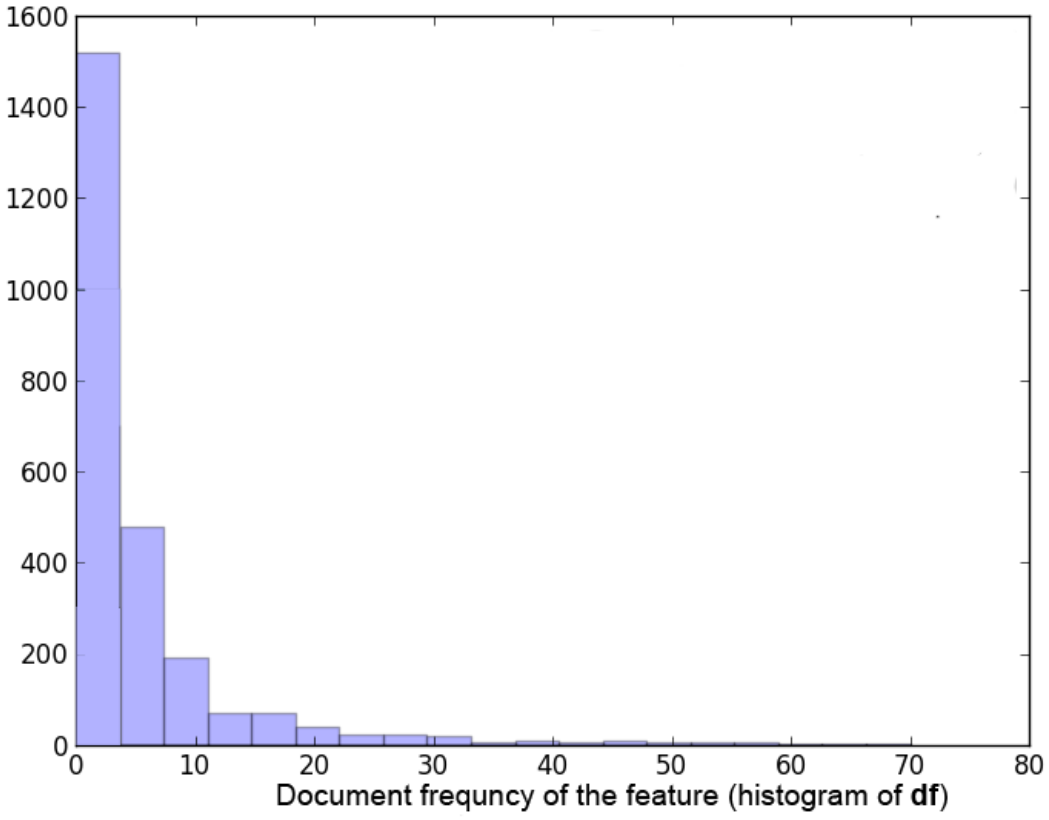

Figure 3.2: Histogram of word document frequency **df** in Twitter train dataset. We see that the same word occurs in small number of different documents, mostly in less than 10 documents.

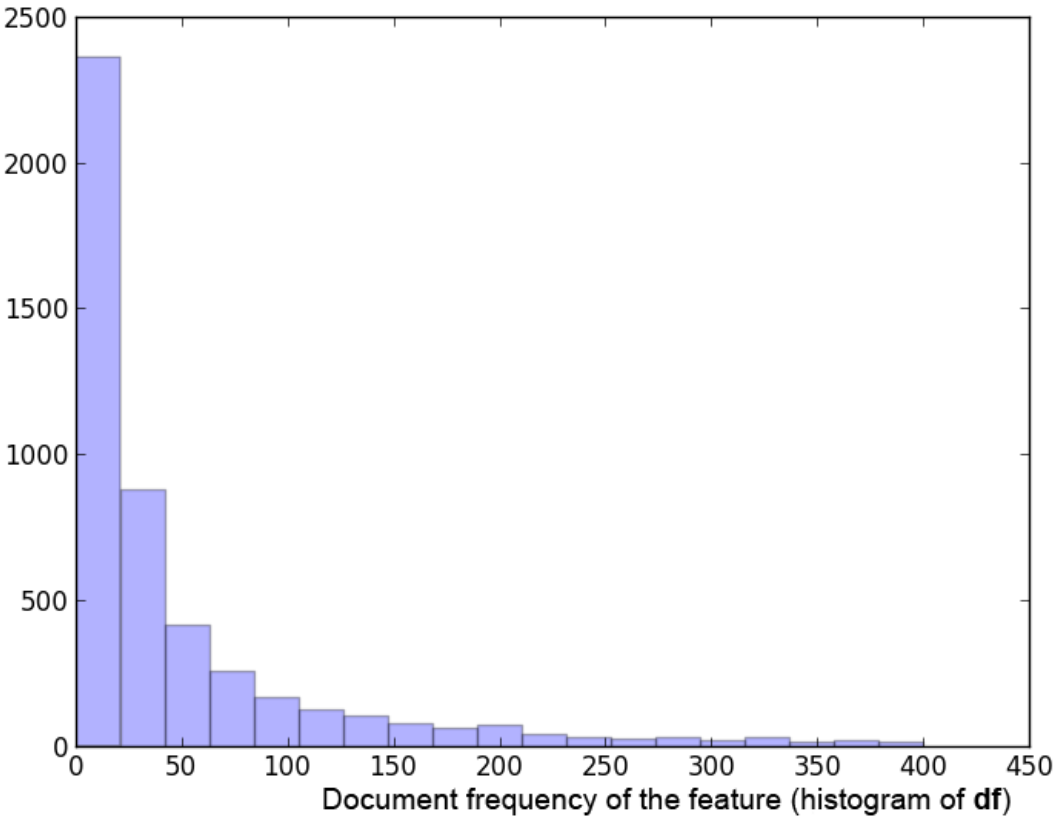

Figure 3.3: Histogram of word document frequency **df** in KDD train dataset. Comparing to Twitter dataset, words appear on average in larger number of documents.

3.2.2 Linguistic Approach for Dimensionality Reduction

Thus far, I have analyzed how well Machine Learning feature ranking methods can perform, including LARS, IG, MI, *etc.*, in reducing the sparsity. However, as I work with the features, having semantic meaning, why not to take into account this information? One of the efficient ways of reducing dimensionality without loosing the information from feature is merging semantically close terms. In this case the term frequency increases as well. In the following subsection I present Word Clustering algorithm, merging words with close meaning and show how feature space significantly can be decreased preserving as much as possible original information.

3.2.2.1 Word Clustering by SubSequence Matching and WordNet lexical database

In formal long text there are many words, having close meaning, but different inclination, tense and so on. Semantically these words can be very similar. However neither lemmatization, nor stemming cannot always result in the basic form, which is the same for all semantically close words. Obviously, these words can have different roots. For some forms the problem of missing possible merge exists. For example, we want to merge words "went" and "go", but we did not merge them, because lemmatization did not result in the same form. Lemmatization and stemming algorithms are quite diverse in handling the affixes. Sometimes, stemming cuts the word rigorously, grouping together completely different words ("genes" and "generally") or not grouping similar words. There are several versions of stemming algorithms: Porter, Lancaster Stemmer[5]. Lancaster stemmer usually makes the word shorter than Porter algorithm. On the other side, lemmatization can be too gentle in keeping affixes of the word and not being able to group similar words by their lemmas. Lemmatization is more gentle way of normalizing words before clustering them and we use it as the first step.

The description of the projects, written by teachers, proved to contain a lot of words with the common root or belonging to similar groups of words. The following 2 clusters of words ["every", "ever", "everyday", "everything", "everyone", "everywhere", "everybody"] or ["create", "creating", "creativity", "created", "creates", "creation"] give an intuition how words can be clustered.

I should note, that by clustering here we mean merging the words together, to be precise, merging their frequencies into one feature. Not all the

[5]http://www.nltk.org/api/nltk.stem.html

words are assigned to a cluster.

Before Word Clustering algorithm is described, the criteria for two words being merged are provided:

1. The words have the same first four letters. With three letters it is more likely to make a mistake, because some words have the same roots. For example, "genre", "gender", "generic" words could be clustered together by mistake.

2. The ratio of string similarity, calculated by SequenceMatcher, should be more than predefined value.

3. The length of 2 words does not have to be equal to avoid words where one or couple of letters are different. For example, "prime" - "price".

4. Words need to have either the same stem (this works well for verbs and inclinations) or semantical meaning.

In my experiments SequenceMatcher module for Python was used. It finds the longest contiguous matching subsequence that contains only informative elements. The module is based on the Ratcliff and Obershelp algorithm [28]. The similarity of two strings is computed as the number of matching characters divided by the total number of characters in the two strings. Matching characters are counted in the longest common subsequence plus, recursively, matching characters in the unmatched region on any side of the longest common subsequence.

The idea of using WordNet tool for finding the level of similarity between the words was postponed for a long time and a number of experiments with language independent methods of word clustering was done. Brown [6], LDA [5], JST [17] clustering techniques are able to find clusters of the words, grouped by the topic. In practice, making relevant clusters, that have words consistent within the cluster according to the topic, is quite hard. There are several reasons for that: we do not know in advance the number of topics, some of the parameters need to be properly tuned. Moreover, words within one topic can differ a lot, and merging the words can "confuse" the classifier.

That is why, I decided to make accurate type of clustering, where only words having the same meaning are grouped. Words from different parts of speech, like "noun", "verb", *etc.*, but the same root or meaning should be merged. In this way dissimilarity within the merged features takes place only in rare cases and meaning is preserved most of the time.

Two versions of clustering algorithm were developed: with and without WordNet. My first trial of making the clustering without WordNet was

good enough, however the problem with the words, having similar roots but not meaning appeared quite often. That is why, WordNet library was incorporated later, that helped to improve the algorithm significantly. The pseudocode for the improved version is presented in Algorithm 2. Only the part where two words are compared before merging is explained. We should note that not all of the words are presented in the WordNet, that is why SequenceMatching ratio at first is checked. Empirically it was found that close enough words (not exactly with the same meaning), should have ratio more than 0.4. Stemming helped to group inflected words and verbs without checking their similarity in WordNet. Words having the same stem, but different meaning are more likely to have shorter stem. That is why condition $w_1[0:4] = w_2[0:4]$ for matching first 4 letters partially mitigates this problem. Random coincidence of the first four letters occurs less often. In case, at least one of the three conditions is not met, I check WordNet similarity. Experimentally it was found, that words having very close semantical meaning have coefficient of similarity more than $\beta = 0.95$. At the same time, for words having similar stem we decrease the coefficient of similarity up to $\alpha = 0.4$ in order to merge words from different parts of speech. It should be noted that all merged words were checked for being grammatically correct by Python enchant tool[6]. They did not contain any numbers or names.

KDD essay data features were merged using this approach. Basic clustering algorithm without WordNet has found 1800 groups of words in improved Word Clustering. In Table 3.2, some of clusters are presented in the first column. Clustering managed to reduce the dimension of the feature space from 5090 to 3048 features, that was quite significant reduction. Improved clustering managed to reduce the vocabulary size to 3192 features. As we see from the Table 3.2, where I compared the result of clustering similar groups of words, some of the irrelevant words were replaced by synonyms instead. Overall quality has been improved quite significantly.

In Sentiment Analysis papers, clustering semantically close words was not researched much. Sparsity problem was not studied well enough, except one of the papers, that was one of the most inspirational for me [33]. In addition to words, the clusters were constructed using Joint Sentiment Topic Model (JST). These clusters served as additional features, but did not replace original features. Naïve Bayes classifier was used for prediction. I followed other approach, that shrinks the size of vocabulary both decreasing the amount of overlapping features and reducing the sparsity by making less zero values.

To conclude, I have presented the algorithm that dealt with sparsity in a very accurate way. Later I show how it helped to increase the performance of

[6]http://pythonhosted.org/pyenchant/tutorial.html

Algorithm 2 Semantic Word Clustering using WordNet

procedure SEMANTIC WORD CLUSTERING(w_1, w_2) ▷ Two words, that are checked to belong to the same cluster
  if $SubsequentMatching(w_1, w_2) > 0.4$ *and* $w_1[0:4] = w_2[0:4]$ *and* $|w_1| <> |w_2|$ **then** ▷
    if $Stem(w_1) == Stem(w_2)$ **then**
      return True ▷ Same cluster
    else
      if $WordNetSimilarity(w_1, w_2) > \alpha$ **then**
        return True ▷ Same cluster
      end if
    end if
  else
    if $WordNetSimilarity(w_1, w_2) > \beta$ **then**
      return True ▷ Same cluster
    end if
  end if
  return False ▷ Different cluster
end procedure

Clustering with SequenceMatcher	**Improved WordNet Clustering**
county, count, counting, countless, counter	count, counting, counter
happen, happy, happening, happens, happened	happens, happened, occur, occurs, occurred, occurring
present, presented, presenting, presenter, press, preserve, presence, presently	presentation present, presented, presenter, presence, presently
generation, generally, generate, generated, generational	general, generation, generous, generally, generate, generated
hand, handle, handed, handicap, handing, handout, handy, handling	hand, handed, handing
home, homework, homeless, homeroom, hometown, homey, homemade	[homeless, homelessness],[home, hometown]

Table 3.2: Comparison of clusters made by two proposed methods

ELM. In the next section, I show more experimental results on sparsity and try to analyze how Machine Learning FS approaches choose the features.

3.2.3 Analysis of tf_idf and its affect on sparsity

Once we have seen that original data is quite sparse, we can access how our observed Feature Selection methods can decrease the sparsity, to which extent and how decreasing the sparsity affect the performance.

First of all, I make the plot of the sparsity for original dataset on Figure 3.4, before Feature Selection. It is a straight line, as the sparsity is calculated on whole set of features. Then features ranked by LARS are added one by one and the sparsity with chosen number of features is computed. In the end all plots meet in 1 point, as all features are taken into account. That gives us the intuition about the "strategy" of the classifier. Except LARS, I also compare **tf_idf** and supervised version of it.

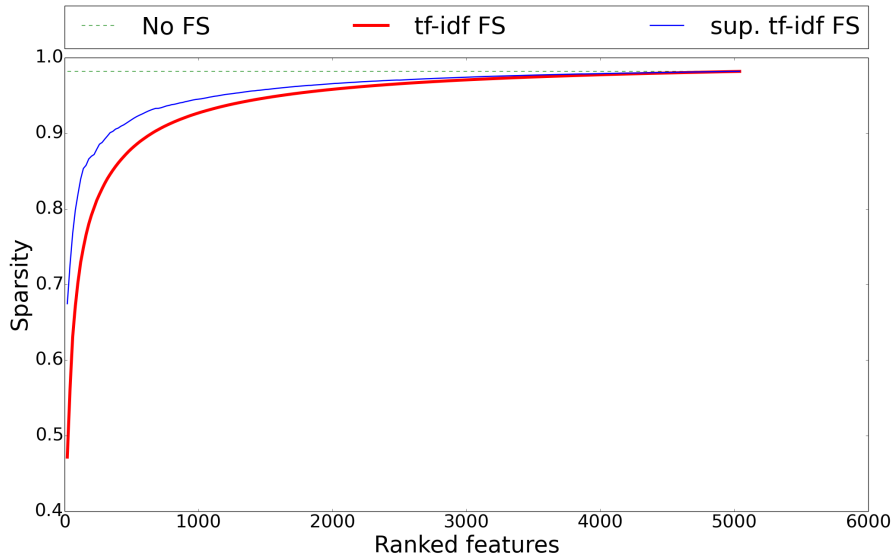

Figure 3.4: Sparsity before WordNet Clustering on KDD Essay train dataset.

What we can observe from the plot is that **tf_idf** selects the features with high frequency, reducing the sparsity significantly. On Figure 3.5 I plot the sparsity, defined in 2.4.1, after applying Word Clustering with WordNet. Although all FS reduce the sparsity **tf_idf** significantly outperforms in this task. The original sparsity coefficient is 0.98. With 500 features, for instance, sparsity coefficient is 0.86 comparing to 0.91 for supervised **tf_idf**. The decrease of sparsity of it is very significant, comparing to supervised FS as well. Supervised version tries to find the terms, that appear in the documents from

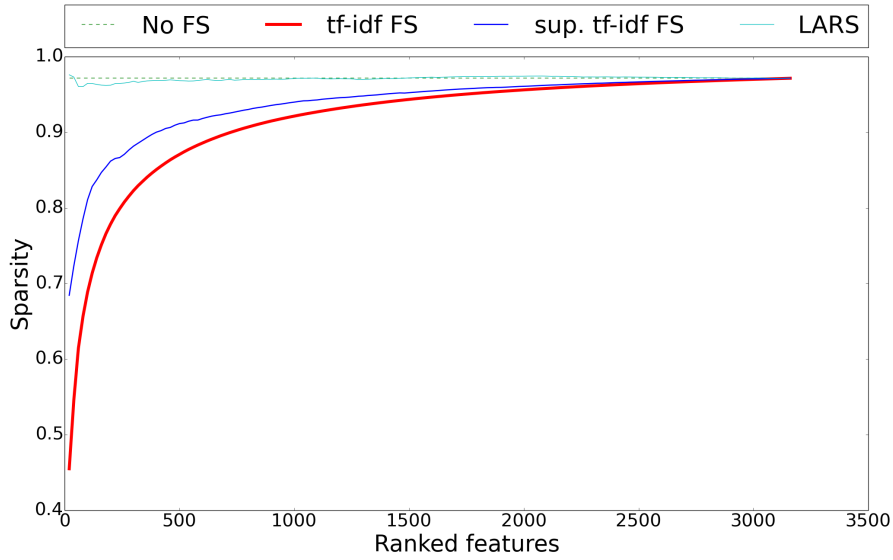

Figure 3.5: Sparsity after WordNet Clustering on KDD Essay dataset. Notice, that after Word Clustering the vocabulary size has decreased up from 5090 to 3192 features, so that baseline sparsity, the sparsity of the dataset with all features, decreased from 0.98 to 0.97.

a certain class. This strategy is not directly connected with the frequencies, however the document frequency is taken into account.

It is important to compare our sparsity plots with the performance of the classifier, that uses these features. Low sparsity is not the guarantee of the "success" and there is no direct relation between it and performance. However, as long as irrelevant features are removed and some of the features are combined into one without significant loss of information, we achieve our goal. Many sophisticated classifiers face the Curse of Dimensionality problem [4] at some point. That is why, to show the benefit for this type of classifiers, I have chosen ELM. At the same time, ELM are quite powerful in sense of handling relatively high dimensional data and there is a potential of wider usage of them in Text Classification problems.

On Figures 3.6 and 3.7 the accuracy of the ELM classifier on the KDD test set is presented. For taking into account the sparsity, I have made plots consistent with previous sparsity plots and the accuracy was made on the ranked features by LARS, **tf_idf** and supervised **tf_idf**. For less than 300 features, supervised version beats unsupervised **tf_idf**, from around 300 features to 1000 performance is mostly higher **tf_idf**. After 1000 features unsupervised version totally outperforms supervised one. One of the reasons is that the amount of features that label information can help to find limited number of

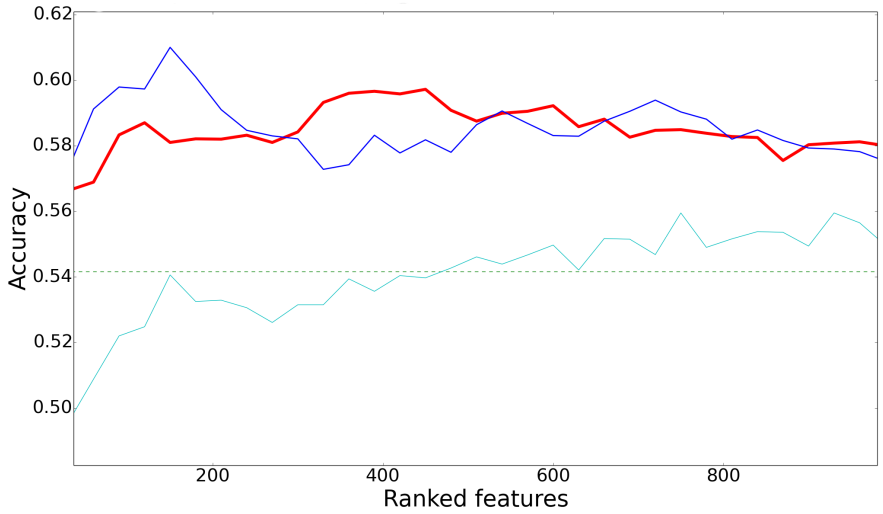

Figure 3.6: The accuracy for ranked features up to 1000 ranked features on KDD Essay test dataset

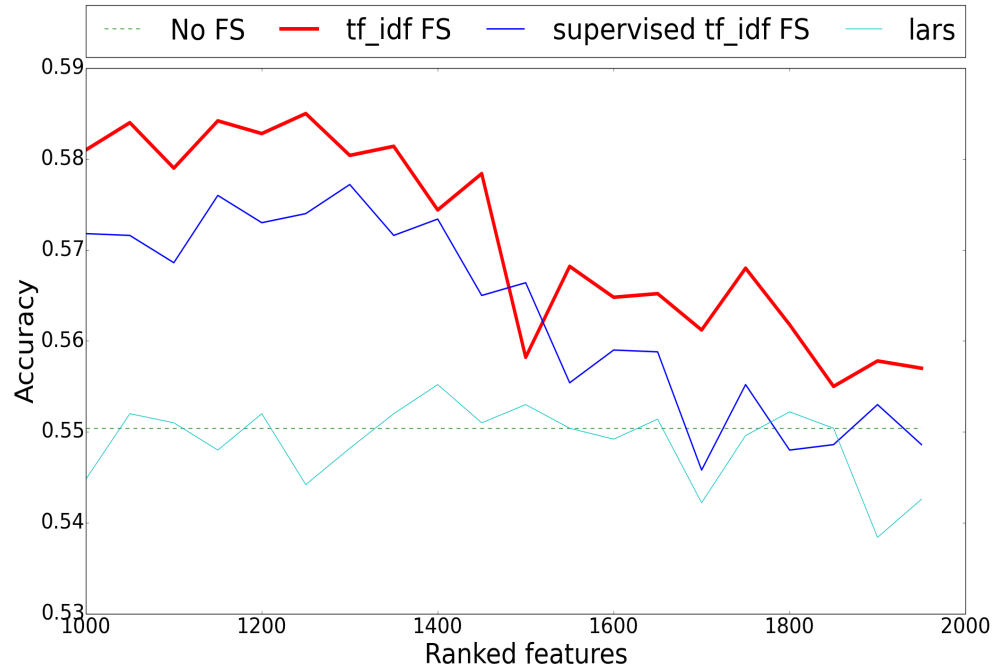

Figure 3.7: The accuracy for 1000 to 2000 ranked features on KDD Essay test dataset

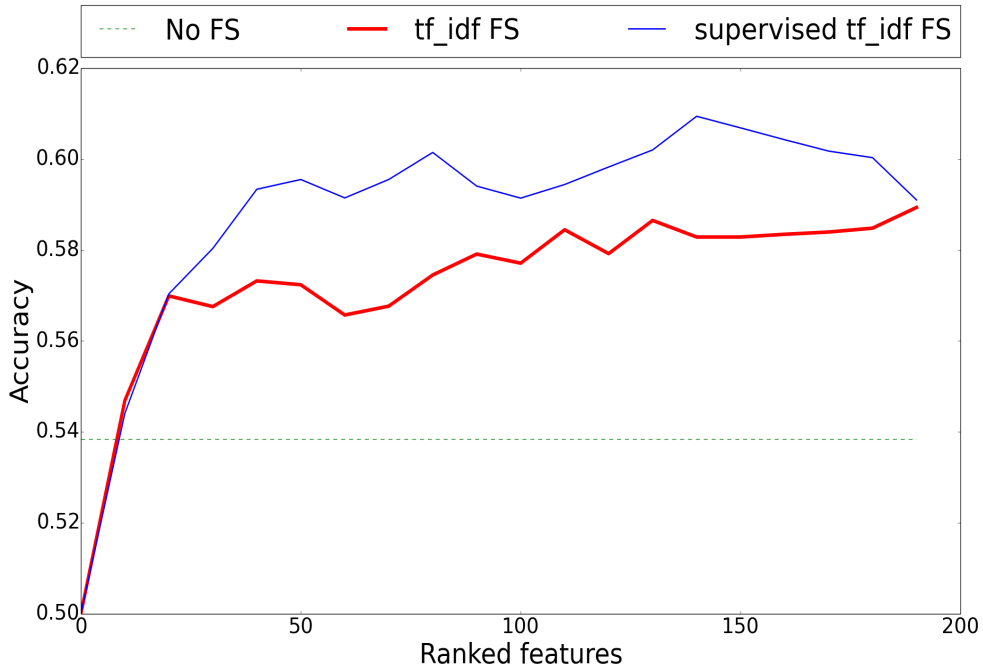

Figure 3.8: The accuracy for ranked features up to 200 ranked features on KDD Essay test dataset with averaged over 20 ELM results

features, that occur in certain type of documents. Many words occur more a less equally in both exciting and not exciting projects. Hence, features, ranking in the beginning can perform quite well, however features ranked in the end can be not relevant or have low frequency. On the other hand, one of the reasons why unsupervised **tf_idf** was better after 1000 features is because it basically takes into account the document frequency of the word, that can be beneficial. In [30] document frequency was proposed as additional constraint for Feature Selection. Nevertheless, the highest accuracy is achieved with supervised **tf_idf** on small number of features.

In order to have a closer look at the performance I made similar experiment, but have restricted feature size up to 200 and run averaged ELM performance over 10 times for each feature. The results are shown on Figure 3.8 respectively. For less than 300 features, supervised version beats unsupervised **tf_idf**, after that the performance is higher for unsupervised **tf_idf**.

Surprisingly, but LARS, that had the highest sparsity going closely to the baseline, performed in a similar way on the test dataset. The accuracy is in average lower than for other FS, but fluctuating along some interval. On the other hand, both **tf_idf** and supervised **tf_idf** accuracies tend to go down

from several hundreds of features. As the sparsity goes up, the accuracy goes down.

In order to understand deeper the strategy of LARS and **tf_idf** based FS, I plotted the histograms of word-distribution from above-described **df** vector. It shows the number of terms, chosen by the FS on the y-axis, occurring in a certain number of documents on the y-axis. We can notice from histogram what features FS favors and what happens, as the number of ranked features grows. Moreover, it helps to predict possible weakness on the test test.

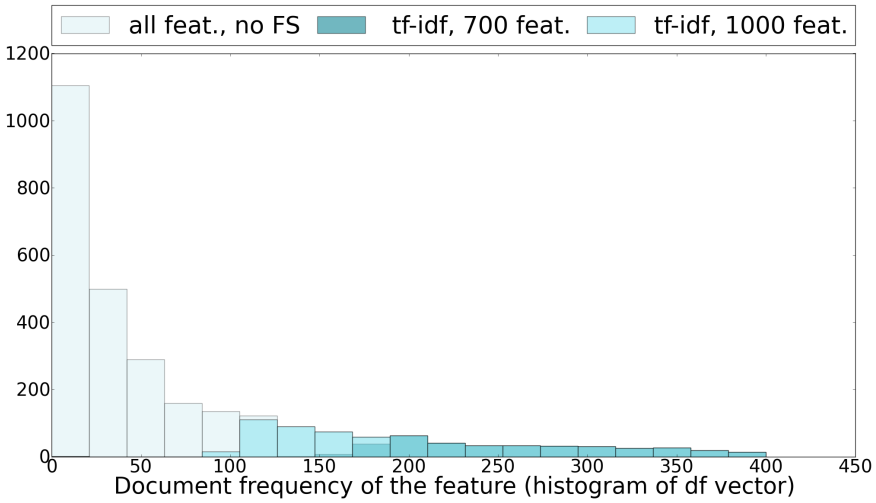

Figure 3.9: Histogram of word distribution for **tf_idf** FS on KDD Essay train dataset. This approach favors the features having high frequency. Ranking of the features made nearly by document frequency. On the plot I limited document frequency up to 400. In fact top ranked words have document frequency up to 2000.

On Figure 3.9 the histogram is plotted for different number of selected features, chosen by **tf_idf**: 700 and 1000. It was surprising to find that it selects exactly the features, occurring in the number of documents belonging to some interval. For example, for 700 features, this interval is between 150 and 500 documents. The less features it selects, the more frequent ones it takes. No one of the features was chosen from too low or frequent number of documents. Therefore, the strategy of this Feature Selection is to select those features, occurring in the middle range of documents, but shifts towards high frequency with smaller number of features. Let us recall the formula of **tf_idf**. It includes not only term frequency, but inverse document frequency, finding the terms having high frequency, but in certain documents. It favors "middle frequencies". They are regarded as more informative and more likely

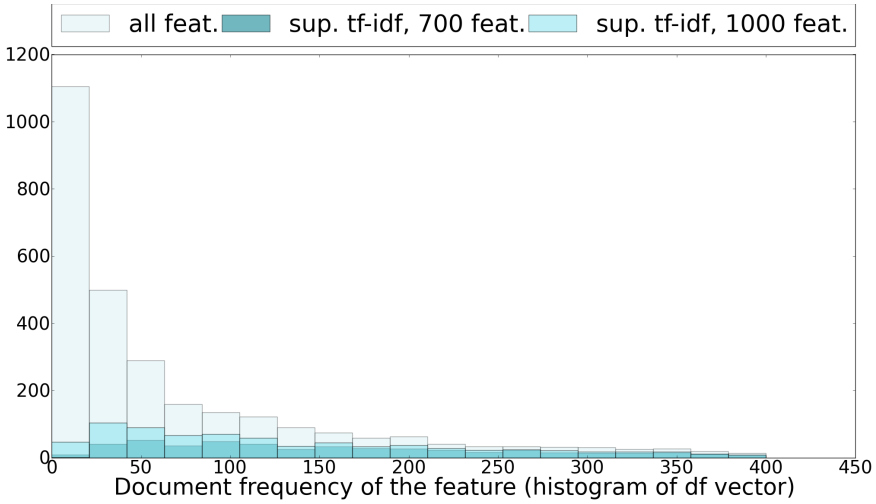

Figure 3.10: Histogram of word distribution for supervised **tf_idf** FS on KDD Essay train dataset. Supervised approach unlike unsupervised takes into account not only the frequencies but also labels. We see that not all the features with high document frequency **df** are selected.

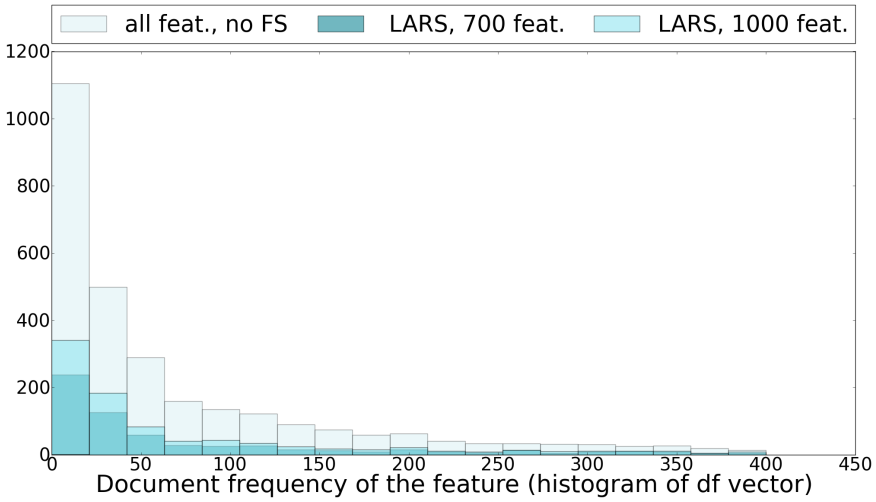

Figure 3.11: Histogram of word distribution for LARS FS on KDD Essay train dataset. It seems that in LARS the selected features are chosen more a less proportionally to the initial document frequencies. We can observe that the distribution of the chosen features seems to be scaled version of the original histogram for the whole dataset. That means that frequency is not the first criteria here.

to occur on unseen data.

For supervised version similar histogram was built on Figure 3.10. Selected features are spread along the document frequency and the strategy is based on those words that appear more in certain class of documents.

The results for similar experiment for LARS were even more surprising. On Figure 3.10 we can observe, that ranking of the features starts more with low frequency words. Selected features seem to be quite sparse, that agrees with our initial sparsity plot. In fact LARS way of selecting the features works similar to stepwise regression. Output is determined by a linear combination of a subset of potential features. In practice with high-dimensional data, where variables are dependent from each other and with the presence of noise the relevance of chosen variables is not proven [11].

In the next section we compare widely used Naïve Bayes in Sentiment Analysis, analyze how beneficial FS is for it.

3.2.4 Naïve Bayes Classifier

One of the models, being popular in Sentiment Analysis, is considered to be Naïve Bayes Classifier (NB), that can handle large high-dimensional problems [25], [41]. Unfortunately most of the classifiers can not handle large dimensions, like in text classification, sometimes with more than 50 thousands of features. That is why we can not compare them their performance with Naïve Bayes Classifier. However it is possible to compare the performance of Naïve Bayes Classifier and ELM on the reduced feature space.

Naïve Bayes Classifier is one of the simplest model, relying on the assumption of variables independence and predicting the class giving maximum a posteriori (MAP). In Text Processing Multinomial Naïve Bayes is used, as each term in the vocabulary is regarded as a separate variable. The probability of a document d of being in class c is computed as:

$$P(c|d) \propto P(c)P(d|c) = \prod_{k=1}^{M} P(t_k|c)\,, \tag{3.1}$$

where M is the number of terms t_i in the document and $P(t_k|c)$ is the probability of occurring term t_k in the document. Conditional probability of a term being in a document with class c is simply calculated as frequency of the term in documents from class c, divided by the number of different words in these documents. Prior probability of class c $P(c)$ is also calculated as the proportion of documents in class c to total number of documents. As we make an assumption about terms being independent the probability $P(d|c)$ can be represented as the multiplication of $P(t_k|c)$.

The prediction of the class is given by MAP estimate:

$$c_{map} = \arg\max_{c} P(c|d). \tag{3.2}$$

If some term from the vocabulary have $P(t_i|c) = 0$ for all classes, meaning that it did not occur in the training set, then $P(c|d) = 0$ for both classes. In order to solve such cases Laplacian Smoothing has been introduced:

$$P(t_k|c) = \frac{N(t_k, c) + \alpha}{(\alpha + 1)\sum_{k=1}^{M} N(t_k|c)}. \tag{3.3}$$

Here α is the smoothing parameter. In case $\alpha = 1$, the probability of the both classes for unseen word is equal. α parameter in Sentiment Analysis papers is often chosen to be 1 [24].

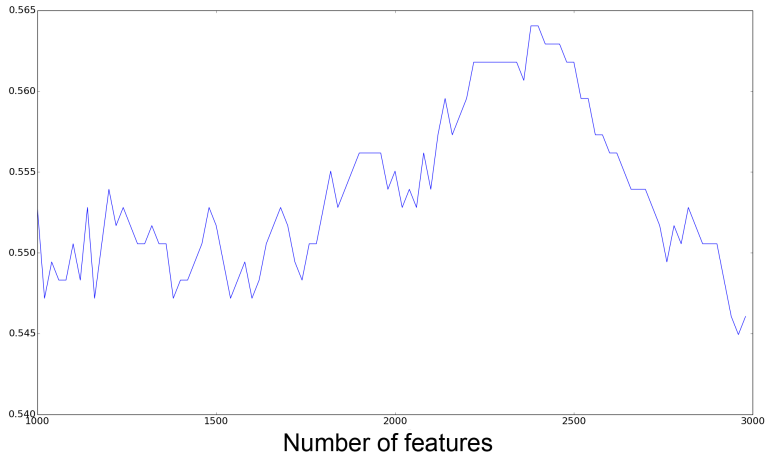

Figure 3.12: The performance of NB with Laplacian Smoothing, $\alpha = 1$, on KDD Essay validation dataset. Maximum accuracy of 56.39% is achieved with 2380 features. Using 2380 features on the test set gives 59.8%. However we still plot the performance with different number of features to see the whole picture on different number of features.

3.3 Classification using Project Data and Comparison with KDD winners' results

In this section I shortly present the details of KDD Project Excitement Prediction challenge, how the results were evaluated and describe the strategy

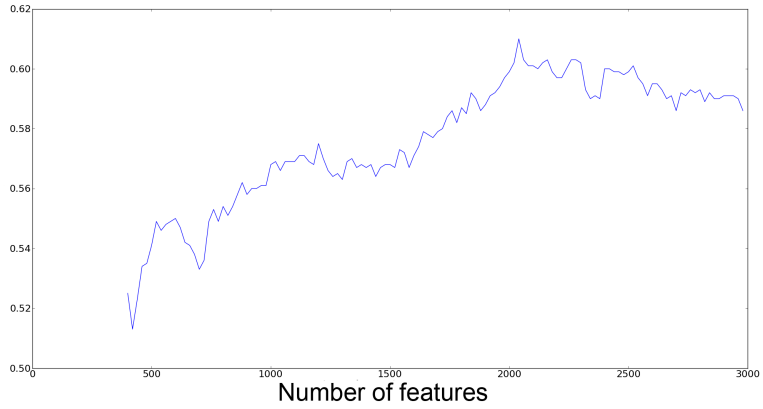

Figure 3.13: The performance of NB with Laplacian Smoothing, $\alpha = 1$, with maximum of 60.93% of accuracy on KDD Essay test dataset. The performance goes down after 2000 of features. The difference in the accuracy between all features 58% and restricted set 60% is significant. 59.8% of accuracy achieved by NB with 2380 features is a good result comparing to maximum accuracy of 60.93%, that we can achieve knowing in advance optimal number of features for NB.

of the winning teams. In the end I try to compare their results with our performance on the ensemble model with project and essay data.

Submissions are evaluated on area under the ROC curve[7] between the predicted probability that a project is exciting. The area shows how good are the results compared with random guess using the True Positive and False Positive Rates. The prediction is made for 44,772 test samples.

Due to the fact that training dataset did not contain essay data for all projects, prediction can not be based on purely essay data for competition prediction. In competition teams could use essay data for some samples only. However I will compare winners' AUC score with my own train and test set, that I described in 3.1.2. First of all, their dataset is highly unbalanced. Secondly, a limited number of submissions is possible. Moreover, my dataset consists of samples having both essay and project data.

Best submission is a mixture of Gradient Boosted Trees. The general idea is to compute a sequence of (very) simple trees, where each successive tree is built for the prediction residuals of the preceding tree [13]. Over the past few years, this technique has emerged as one of the most powerful methods for predictive data mining [8].

[7]https://www.kaggle.com/wiki/AreaUnderCurve

[8]http://www.statsoft.com/Textbook/Statistics-Glossary/P/button/p#Predictive

Winning teams used the history of the previous donations for the same teacher. In addition, they have chosen for training set different time periods, building the ensemble models with different time period. Not only the information about teacher donation results was important, but also teachers who donated to exciting projects have a higher likelihood to post exciting projects. Feature engineering was done quite extensively by many teams. Some of the donation history variables were adjusted according to the population mean. To account for time trend, they had features called $\mathbf{avg - X - prediction}$ where X was a sliding window of biweekly/monthly/bimonthly predictions from an initial model. For text, they have created features by using logistic regression with **tf_idf** score.

Submissions are evaluated on area under the ROC curve between the predicted probability that a project is exciting. For each project in the test set, one should predict a real-valued probability that the project was exciting.

The maximum AUC probabilistic score I got for test set was 68.58%. I have presented the results, calculated by Decision Tree Classifier[9], implemented in sklearn python library. Except Decision Tree I have tried to make experiments with ELM, SVM, KNN, Random Forests Classifiers available in sklearn library. The best performance I could achieve with Decision Trees. I should note, that Decision Trees are from the same family of algorithms as Random Boosted Trees, that is why it is relevant to show the performance of it. Basically I have transformed all the categorical features to binary representation and concatenated with other binary and numerical data. As a result 70 dimensional matrix was constructed. The projects in training, validation and test sets were the same as in Essay Data: 4000 projects in the training set, 890 in validation and 1000 in the test set. I have conducted many experiments, trying to use ELM, Random Forest Classifier, KNN Classifier and their averaged ensembles. As these classifiers are not in the scope of the thesis, I skipped presenting results, except the best one for this data. Decision Trees for Project Data solely outperformed other classifiers. I tuned **min_samples_split** parameter to 400. Other parameters were set to their default values. The winner of the Kaggle competition got 67.814% with the probabilistic outputs. Of course, my results were calculated on my own test set, however it gives approximate comparison. I should note, that the performance was solely based on the project features, as the performance did not improve by adding textual data. On Figure 3.14 I have shown the AUC score of my classification results. From there we can notice, that after adding one feature, the score has risen up to the highest score of 68.62%. This feature

Data Mining

[9]http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

is the cost of project fullfilment.

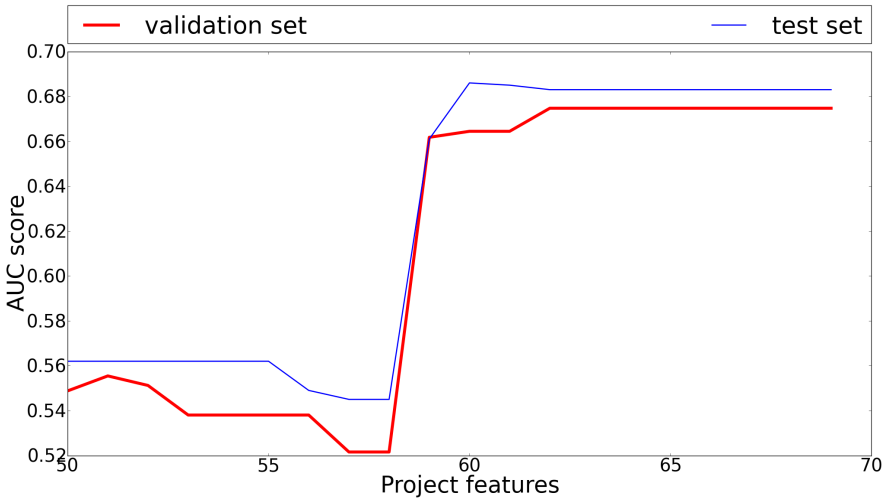

Figure 3.14: The performance of Decision Tree Classifier on KDD Project validation and test datasets. Maximum AUC score of 68.62% on the test set is achieved with 60 features. There is a significant increase of the score after adding feature, responsible for the cost of fulfillment. This feature was also chosen during Forward Selection procedure.

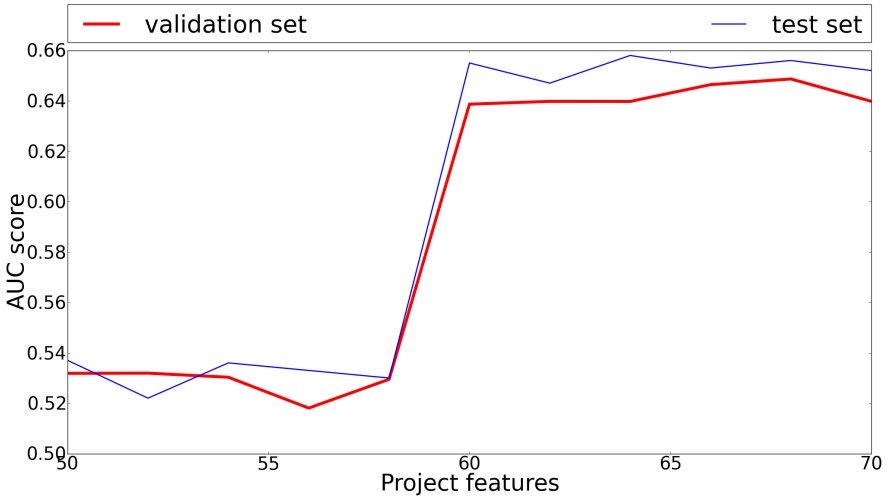

Figure 3.15: The performance of ELM Classifier with 60 neurons in hidden layer on KDD Project validation and test datasets. Maximum AUC score of 65.6% on the test set is achieved with 64 features. The results are averaged over 20 runs of ELM for each feature.

Despite the fact that Decision Trees outperformed other methods on low-dimensional project data, ELM outperformed Decision Trees on sparse high-dimensional essay data. In the next section I make the analysis of the top relevant words, chosen by Feature Selection and finally get some intuition about the data and what projects had high probability of receiving the funding.

3.4 Comparison of the selected features

Although I have discussed a lot the results of the Feature Selection methods, I actually have never seen what words are selected in each of the methods and what meaning they carry. I made some analysis of the top ranked words by LARS and both supervised and unsupervised versions of **tf_idf**.

On the Figures 3.16 and 3.18 I show a few most important words, selected by **tf_tdf** approach before the Word Clustering in KDD Essay dataset. By Word Clustering I mean applying Improved Word Clustering using WordNet that reduced vocabulary for essay representation. Similar results were obtained before and after Word Clustering. You can see them on Figures 3.17 and 3.19. The size of the word on the figure correlates with the rank of the word.

I tried to investigate what is the meaning behind the top words chosen by making a search with these words via the donorschoose.org website. For that I searched the projects containing a specific key word and browsed quickly in what sentence this word is used plus what is the general idea of the project. Surprisingly, but in most of the cases for me it was clear from one sentence why the word was significant, especially when the word referred to some item required.

The word "hour" on Figures 3.16, 3.18, 3.17, 3.19 often was used in the context of "Genious hour" concept[10]. The idea comes from the fact that pupils quite rare do the things that they really want in the class. The concept of "Genious hour" means making sessions where pupils implement some projects, related to their interests. It gives a chance to try something new, exciting and possibly get the inspiration for future work in this area. Many projects mentioning this concept, require e-books, computers and other devices and tools. The word "house" on Figures 3.16, 3.18, 3.17, 3.19 mostly corresponds to the series of books, called "Magic Tree House", that are popular adventurous books written by American author Mary Pope Osborne. The word "relaxed" on Figures 3.16, 3.18, 3.17, 3.19 was one of the popular

[10] genioushour.com

terms for asking some comfortable furniture, so that pupils can be relaxed and enjoy reading books on the comfortable furniture. "My students sit on hard chairs and write on wobbly desks", - wrote one of the teachers. They want to make their classroom a cosy place and to encourage pupils to love reading. In the description of the students, teachers exploit word "mixture" shown on Figures 3.16, 3.17 often to emphasize the diversity of students, studying in the same class. Student can have different grades, capabilities, financial situation in the family. At the same time a teacher wants to give all the students equal opportunities for studying by buying the tablets, *etc.* I have noticed, that tablets are one of the most asked items. There are some amount of projects, requiring tools for "mocking" crime investigation, particularly, kit determination of unknown substances. The word "mock" is shown on Figures 3.16, 3.18, 3.17, 3.19. Moreover, teachers are interested in making practical sessions of "mocking" the interview or "mocking" writing the CV.

The word "jacket" on Figures 3.16 and 3.18 was mostly used as a "jacket for e-book or Ipod". Except that, it referred to "life jackets", "uniform for school", "jackets", "cooking jacket", *etc.* One of the topics, that seems to be well supported is a "theatrical" performance shown on Figure 3.18. Indeed, many of the projects in this area seemed to be quite unusual and exciting. Different kinds of stage equipment and costumes are needed. I have checked the number of donation in the current projects and saw that these types of projects are getting the donations quicker.

For comparing the words, 3 approaches were chosen: LARS, supervised and unsupervised **tf_idf**. We can notice that both versions of **tf_idf** were able to rank higher concepts, related to the asked resources or some situation in the class. All of the words described gave a clue about the project. For example, "jacket" meant mostly Ipad cover, "house" - the series of books, *etc.* At the same time, both **tf_idf** versions have good amount of intersecting words, that are shown on the Figures 3.20 and 3.21, including important concepts, such as "introduce", "relax", "hour", "house", *etc.*

Features selected by LARS are shown on the Figure 3.22. Most of the words do not correspond to some topic, instead they are mostly adjectives and verbs: "invaluable", "conscious", "inquisitive", "exceptional", "dirty", *etc.* Here comes the question, do we need adjectives that are not corresponding to certain topic and not holding other meaning . At the moment this question is open. Selected word "hour" showed that if the word expresses phenomena, for example "Genious hours", then no matter what part of the speech it belongs to, it can be very crucial for classification. Except that, unlike in Sentiment Analysis, adverbs with emotional meaning seem to have less significance. Classification performance was much lower with features ranked by LARS than using **tf_idf** approach.

Figure 3.16: Top selected words by **tf_tdf** approach in KDD Essay dataset before Word Clustering.

Figure 3.17: Top selected words by **tf_tdf** approach in KDD Essay dataset after Word Clustering. We can notice that after clustering the many words stayed the same. Word "coat" is corresponding to word "jacket", that was chosen before clustering.

Figure 3.18: Top selected words by supervised **tf_tdf** approach in KDD Essay dataset before Word Clustering.

Figure 3.19: Top selected words by supervised **tf_tdf** approach in KDD Essay dataset after Word Clustering. Most of the important words has not changed comparing to previous figure.

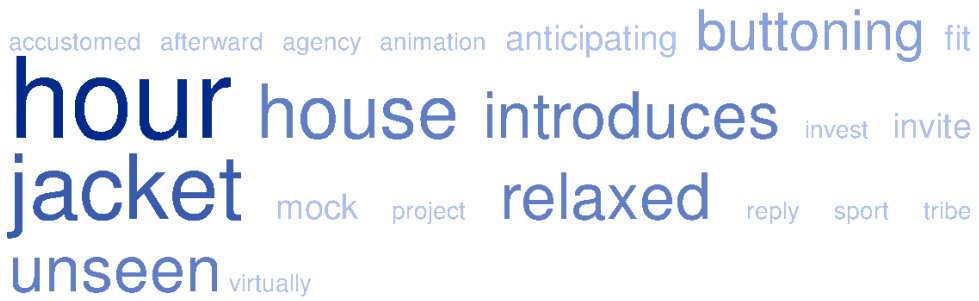

Figure 3.20: Top common words selected by supervised and unsupervised **tf_tdf** in KDD Essay dataset before Word Clustering.

Figure 3.21: Common words selected by supervised and unsupervised **tf_tdf** in KDD Essay dataset after Word Clustering. Top common words before and after Word Clustering are nearly the same, except stem replacements, like "buttoning" - "button", "virtually" - "virtual".

Figure 3.22: Top selected words by LARS feature ranking in KDD Essay dataset after Word Clustering. The words selected do not represent items requested, instead more verbs and adjectives were chosen.

Chapter 4

Conclusions

In this thesis I observed the importance of sparsity reduction in Polarity Classification task. By sparsity reduction I mean selecting the most important features and merging semantically similar ones. It was shown, that sparsity can be decreased by:

1. Choosing the terms, groups of words for word count representation, that are highly probable to be significant. For example, I used along with frequent unigrams and bigrams, sentiment dictionary words, that contain already "emotionally colored" words. For Twitter dataset vocabulary consisted of Negations, Slang words and Emoticons along with unigrams and bigrams. Ranking words within the group can be done by Information Gain, for example. Bigrams are chosen by Pointwise Mutual Information.

2. Careful preprocessing of the text, handling popular mistakes. In Twitter, repetition is one of the wide-spread type of mistakes. The algorithm to handle them was presented.

3. Working with lemmas, instead of stems to avoid possible word disambiguation.

4. Efficient feature ranking by supervised **tf_idf** score. Despite of the fact, that this score helps to make feature ranking, word count representation outperformed **tf_idf** weights representation on ELM.

5. Merging the words with similar semantical meaning. I provided Word Clustering algorithm for that, using WordNet library.

Polarity Classification problem was investigated on two datasets: SemEval 2013 Twitter Sentiment Analysis and KDD Project Excitement Prediction. In Twitter dataset the sentiment and in KDD the project excitement

had to be predicted. KDD challenge is based on the essays from the website, where teachers ask to donate the money for educational purposes. Best performance for both datasets was achieved by using the proposed Word Clustering and supervised **tf_idf** score with less than 250 features on Extreme Learning Machine Classifier. The original dimensionality of the KDD dataset was 5029. I was able to decrease the feature size by more than 20 times!

It was shown, that sparsity does affect the performance. Feature Ranking is especially important in high-dimensional feature space. Taking restricted set of ranked features selected by supervised **tf_idf** in most cases outperformed unsupervised version, that was based on the document frequency of the words. Although unsupervised version provided even larger sparsity reduction comparing to supervised version, it did not take into account labels and gave lower accuracy. LARS feature ranking did not show high accuracy and did not manage to select significant features in observed datasets.

Word count data representation was a better choice, than using **tf_idf** weights. I have shown, how NLP methods, like **tf_idf** and Word Clustering, can be used for alleviating the sparsity. There is a big potential for shrinking the data, especially for the classifiers, that suffer from high-dimensional data. Word Clustering that effectively searches synonym words was introduced. Merging the words with very close meaning was able to decrease the feature space significantly. In our case from 5029 to 3182 features for KDD dataset.

Extreme Learning Machine Classifier was used for solving Text Classification problems. I have shown that it can be used for efficient and fast Text Classification with reduced amount of features. We should note, that ELM unlike Naïve Bayes is more sensitive to sparsity. Naïve Bayes handles high-dimensional data high fast, but at the same time also suffers from redundant features.

Last, but not the least, feature ranking gives an opportunity to make important insights about the data by observing top selected features and some projects, having these top selected features. In KDD dataset, I have inducted that teachers mostly asked for donations to support or motivate pupils to read more. For example, by buying interesting series of books, good furniture to read in a relaxed way, Ipads for internet browsing, *etc.* Except that, theatrical projects are getting much donations.

The thesis has started from the interest in Sentiment Analysis. However, at some point I understood the importance of feature engineering and reducing the sparsity, that attracted my attention. One of the most successful tools for Sentiment Analysis SentiStrength, used in many languages, showed that we can not avoid linguistic information while enhancing the quality of prediction. Using prior linguistic information about features is a possible way to

improve data interpretability for a classifier and to make better predictions.

Bibliography

[1] Abbasi, A., Chen, H., and Salem, A. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Transactions on Information Systems 26*, 3 (June 2008), 12:1–12:34.

[2] Baccianella, S., Esuli, A., and Sebastiani, F. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta* (2010).

[3] bin Huang, G., yu Zhu, Q., and kheong Siew, C. Extreme learning machine: Theory and applications. *Neurocomputing 70* (2006), 489–501.

[4] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *The Journal of Machine Learning Research 3* (Mar. 2003), 993–1022.

[6] Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. Class-based n-gram models of natural language. *Computational Linguistics 18* (1992), 467–479.

[7] Chaumartin, F.-R. Upar7: A knowledge-based system for headline sentiment tagging. In *Proceedings of the 4th International Workshop on Semantic Evaluations* (2007), SemEval '07, pp. 422–425.

[8] Christopher D. Manning, P. R., and Schtze, H. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[9] Cortes, C., and Vapnik, V. Support-vector networks. *Machine Learning* (1995), 273–297.

[10] de Albornoz, J. C., Plaza, L., and Gervs, P. Sentisense: An easily scalable concept-based affective lexicon for sentiment analysis. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)* (May 2012).

[11] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. Least angle regression. *Annals of statistics 32*, 2 (2004), 407–499.

[12] Francois, D. *High-Dimensional Data Analysis.* VDM Publishing, 2008.

[13] Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics 29* (2000), 1189–1232.

[14] Hassan Saif, Miriam Fernandez, Y. H., and Alani, H. Evaluation datasets for twitter sentiment analysis: A survey and a new dataset, the sts-gold. In *In CEUR Workshop Proceedings of ESSEM workshop* (December 2013), pp. 9–21.

[15] Keerthi, S. S. Generalized lars as an effective feature selection tool for text classification with svms. In *Proceedings of the 22nd International Conference on Machine Learning* (New York, NY, USA, 2005), ICML '05, ACM, pp. 417–424.

[16] Kirby, G. Zip's law. *UK Journal of Naval Science 10*, 3 (1985), 180–185.

[17] Lin, C., and He, Y. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (2009), pp. 375–384.

[18] Lin, C., He, Y., and Everson, R. Sentence subjectivity detection with weakly-supervised learning. In *The 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)* (2011), pp. 1153–1161.

[19] Liu, Y., Loh, H., Kamal, Y.-T., and Tor, S. Handling of imbalanced data in text classification: Category-based term weights. In *Natural Language Processing and Text Mining*, A. Kao and S. Poteet, Eds. 2007, pp. 171–192.

[20] Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C., and Lendasse, A. OP-ELM: Optimally-pruned extreme learning machine. *IEEE Transactions on Neural Networks 21*, 1 (January 2010), 158–162.

[21] Miche, Y., van Heeswijk, M., Bas, P., Simula, O., and Lendasse, A. TROP-ELM: a double-regularized ELM using LARS and tikhonov regularization. *Neurocomputing 74*, 16 (September 2011), 2413–2421.

[22] Miller, G. A. Wordnet: A lexical database for english. *Communications of the ACM 38*, 11 (1995), 39–41.

[23] Nakov, P., Rosenthal, S., Kozareva, Z., Stoyanov, V., Ritter, A., and Wilson, T. Semeval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)* (June 2013), pp. 312–320.

[24] Narayanan, V., Arora, I., and Bhatia, A. Fast and accurate sentiment classification using an enhanced naive bayes model. In *Intelligent Data Engineering and Automated Learning IDEAL 2013*, vol. 8206. 2013, pp. 194–201.

[25] Narayanan, V., Arora, I., and Bhatia, A. Fast and accurate sentiment classification using an enhanced naive bayes model. In *Intelligent Data Engineering and Automated Learning IDEAL 2013*, vol. 8206. 2013, pp. 194–201.

[26] Nielsen, F. A new anew: evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages* (2011), pp. 93–98.

[27] Pang, B., and Lee, L. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr. 2*, 1-2 (2008), 1–135.

[28] Ratcliff, J. W., and Metzener, D. Pattern matching: The gestalt approach. *Dr. Dobb's Journal* (1988), 46.

[29] Reisenzein, R. Arnold's theory of emotion in historical perspective. *Cognition and Emotion 20*, 7 (2006), 920–951.

[30] Rogati, M., and Yang, Y. High-performing feature selection for text classification. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management* (2002), CIKM '02, pp. 659–661.

[31] Saif, H., Fernández, M., He, Y., and Alani, H. On stopwords, filtering and data sparsity for sentiment analysis of twitter. In *LREC 2014, Ninth International Conference on Language Resources and Evaluation. Proceedings.* (2014), pp. 810–817.

[32] Saif, H., Fernandez, M., He, Y., and Alani, H. Senticircles for contextual and conceptual semantic sentiment analysis of twitter. In *The Semantic Web: Trends and Challenges*, vol. 8465 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 83–98.

[33] Saif, H., He, Y., and Alani, H. Alleviating data sparsity for twitter sentiment analysis. In *CEUR Workshop Proceedings* (2012), pp. 2–9.

[34] Salton, G. *The SMART Retrieval System: Experiments in Automatic Document Processing.* 1971.

[35] Sharma, A., and Dey, S. Performance investigation of feature selection methods. *IJCA Special Issue on Advanced Computing and Communication Technologies for HPC Applications* (2012), 15–20.

[36] Shi, L., Agarwal, N., Agrawal, A., Garg, R., and Spoelstra, J. Predicting us primary elections with twitter. In *Proceedings of the NIPS 2012 Workshop on ocial Network and Social Media Analysis: Methods, Models and Applications* (2012).

[37] Shlens, J. A tutorial on principal component analysis. *CoRR abs/1404.1100* (2014).

[38] Torgerson, W. S. Multidimensional scaling: Theory and method. *Psychometrica 17* (1999), 307–311.

[39] Turney, P. D. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (2002), ACL '02, pp. 417–424.

[40] Usha, M., and Indra Devi, M. Analysis of sentiments using unsupervised learning techniques. In *Information Communication and Embedded Systems (ICICES), 2013 International Conference on* (Feb 2013), pp. 241–245.

[41] Wang, S., and Manning, C. D. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual*

Meeting of the Association for Computational Linguistics: Short Papers - Volume 2 (2012), ACL '12, pp. 90–94.