POLITECNICO DI MILANO

Department of Electronics, Informatics and Bioengineering

M.Sc. Programme in Computer Science and Engineering

Machine Learning Techniques for Social Media Analysis

Advisors:

Prof. Letizia TANCA

Prof. Maristella MATERA

Ing. Riccardo MEDANA

Master Thesis of:

Petar KORDA Matr. 852627

Pavle VIDANOVIC Matr. 854472

Academic Year 2017-2018

We would like to express our sincere appreciation to our supervisors and families for all their support.

Abstract

In today's society everything is happening on the Internet, in particular on social networks. Social networks play central role in everyday life of average people. So naturally, companies recognize the opportunity and try to make use of that by changing their business plans and focusing on potential customers on social networks. Business is realized by means of the presence of the company on web and by producing content that will take customer's attention. In return, users share their opinions about particular products by leaving comments on them and reacting to the company's posts.

Taking this scenario into consideration, it is useful to have automated techniques to analyse user reactions on products that a company is offering. Also knowing types of people fallowing and leaving opinions on products can be turned into advantage for defining future business plans. For example, to predict which products can be attractive for specific user groups or to determine the best time when to lunch products. We recognized the potential of that and that's why we were eager to examine sentiment analysis tools and machine learning algorithms to improve the analysis.

In this thesis we have built an automatic procedure for calculating sentiment of users who comment on specific company's posts properly cleaned by means of intelligent spam filter. Sentiment analysis was done separately on text and on emojis. For evaluating text sentiment, we used open source APIs and for emojis we used tables of evaluation for each emoji. The spam filter was designed using supervised machine learning techniques to determine spam, not just by searching URL patterns in comments, but checking the whole text content. We have also built a clustering module that uses unsupervised learning techniques on user data and visualize characteristics for each discovered group. Finally, in last chapter we define how the defined models and algorithm can be used together in an API to evaluate success of company's posts.

Sommario

Nella società di oggi tutto accade in Internet, in particolare nei social network. I social network svolgono un ruolo centrale nella vita di tutti i giorni. Naturalmente, le aziende riconoscono diverse opportunità derivanti da questo uso diffuso dei canali social e cercano di farne uso cambiando i loro piani aziendali e concentrandosi sui potenziali clienti così come essi emergono dai social network. Parte del business delle aziende è quindi realizzato attraverso la loro presenza sul web e producendo contenuti che cattureranno l'attenzione del cliente. Come reazioni, gli utenti condividono le loro opinioni su determinati prodotti, lasciando commenti su di loro e reagendo ai post della dell'azienda.

Prendendo in considerazione questo scenario, è utile disporre di tecniche automatizzate per analizzare le reazioni degli utenti sui prodotti offerti da un'azienda. Anche la conoscenza dei tipi di follower e degli utenti che lasciano i commenti sui prodotti può essere trasformata in vantaggio per la definizione di piani aziendali futuri. Per esempio, tali informazioni possono essere utili per prevedere quali prodotti possono essere attraenti per specifici gruppi di utenti o per determinare il momento migliore in cui lanciare e presentare i prodotti. Abbiamo riconosciuto il potenziale di questo aspetto ed è per questo che abbiamo esaminato gli strumenti di analisi delle opinioni e gli algoritmi di apprendimento automatico per migliorare l'analisi.

In questa tesi abbiamo costruito una procedura automatica per il calcolo delle opinioni degli utenti che commentano i messaggi specifici di un'azienda, correttamente filtrate mediante un algoritmo anti-spam intelligente. L'analisi delle opinioni è stata fatta separatamente sul testo e sugli emoticon. Per valutare le opinione emergenti dal testo, abbiamo utilizzato API open source; per gli emoticon abbiamo utilizzato tabelle di valutazione predefinite per ogni emoticon. Il filtro anti-spam è stato progettato utilizzando tecniche di apprendimento automatico supervisionato per determinare lo spam, non solo ricercando URL nei commenti, ma controllando l'intero contenuto del testo. Abbiamo anche creato un modulo di clustering che utilizza tecniche di apprendimento non supervisionato sui dati degli utenti e visualizza le caratteristiche per ciascun gruppo individuato. Infine, nell'ultimo capitolo definiamo come i modelli e gli algoritmi definiti possono essere utilizzati in modo combinato in un'API per valutare il successo dei post pubblicati da un'azienda.

Contents

Chapter 1

Introduction

Social media gives businesses an unprecedented opportunity for connecting with customers and prospects. While there are numerous social networks that provide a vast array of tools for offering customer service, explaining how your products work and much more, it is important to realize that simply having a social media presence is no guarantee of success. It is essential to test and track your results so that you can discover the most effective strategies and that is why analytics of social data are so important.

We know that users of social networks represent potential customers, they are also proactive in a sense that they can share their opinion about certain products that company or brand shares. So, in order to grow their business and to increase the profit, companies need to analyse user reactions and behaviour on their products shared on social networks. As a result, a company can have models of the users and can predict weather new product will be successfully accepted and attractive to potential customers.

Analysis of social network content is difficult because conversation on social networks differs in many ways from normal conversation. Contents are enriched with emojis, hashtags, mentions and spams. They need to be cleaned and the raw text needs to be processed to find the information behind it.

By analysing social networks, data about certain products, brands or certain marketing campaigns can be very useful for companies and their business. Companies can predict future trends, increase the profit and thus be in advantage over the competition.

Within previously described context, this project gives the opportunity to companies to analyse contents to determine sentiment of customers on specific products, to use supervised machine learning algorithms within spam filter module to efficiently detect and remove spams from datasets, and finally to exploit a clustering module that discover user groups and their characteristics. The clustering module can be used by companies in a marketing segmentation process, which is very important for creating long-term business plans.

Sentiment analysis mentioned before represents the process of computationally identifying and categorizing opinions expressed in a piece of text, especially to determine whether the writer's attitude towards a topic is positive, negative, or neutral [Oxford dictionary definition]. The sentiment analysis process in this project can be divided into two separate processes. The first one determines sentiment of raw text data using a sentiment analysis API. We used the Vader sentiment analysis API because of its performance and accurate results. The second process considers just emojis filtered from text data and determines their sentiment using predefined emoji-sentiment tables. Finally, those two sentiments are

combined in an optimal way, using weights for text and emoji sentiment, that represents the real sentiment of provided data. It can be used to determine reactions of customers on social networks about specific product.

The spam filter module is one of most important components in every application that focuses on user interactions. It is widely used in e-mail applications, instant messaging applications and in all kinds of websites publishing user generated content. The spam filter we created is equipped with two components. A text processor uses regex expressions to detect links inside the text. It is used in filtering comments without links. Comments with links are classified as spams without further inspection, and the rest of them are inspected with a second component. The second component is more intelligent, and it is implemented as a trained machine learning model that detects spams by checking the text content. Among the supervised machine learning algorithm, we selected Naïve Bayes algorithm because it showed the best performance and represents the best fit for this kind of dataset.

The clustering module is created as an unsupervised machine learning model that finds patterns in user data and discovers user groups. Each user group has specific characteristics that are examined in a visualization module after clusters are determined. This is particular useful in marketing segmentation where company can make strategies on which are specific user groups and how to target them. In that way a company can manage its resources in optimal and more effective way. The K-means algorithm was selected for this purpose and the (optimal number of) clusters are determined using elbow analysis technique.

Finally, the previously described models, combined together, can be used in one complete API which makes predictions on successfulness of company's posts left on social networks by doing analysis on customers data. This API can be used as a component in any company's business intelligence application.

1.1 Structure

This thesis is organized as follows:

- Chapter 2 illustrates the reasons for having tools for analysing social networks within company's applications, the advantage of having them and how to use it. We will mention examples of use.
- Chapter 3 is dedicated to automated sentiment analysis of social media content. Here we will talk about sentiment computation, pre-processing of emoji's and their sentiment, and combining text emoji sentiment. We also describe our approach and architecture.
- Chapter 4 is about a machine learning approach to create a spam filter. This includes pre-processing, training of the dataset and evaluation of classifier.
- In Chapter 5 we present an unsupervised approach to social data analysis. We will go through all steps of this process which include: fetching the data from social networks, pre-processing, unsupervised learning techniques, elbow analysis and visualization of clusters characteristics.

- Chapter 6 is dedicated to describing the API that uses the previously mentioned modules in order to analyse custom social data.
- Chapter 7 finalizes the purpose of this project and present the conclusion. It also describes future improvements of this project.

Chapter 2

State of Art

This chapter describes state of art of analysis of social data using machine learning techniques. Chapter consists of four sections, each of them describing currently used techniques for analysis of social data, emphasising the need of such activities and the impact on the company's business.

1. Applications of social data analytics in various companies
2. Commercial tools used for social data analytics
3. Open source libraries and frameworks
4. Research work

2.1 Applications of social data analytics in various companies

"What other people think" has always been an important piece of information for most companies during the decision-making process.

Social data analytics is one of the enabling technologies that can be applied in many types of companies and fields. Especially the ones that includes marketing and where the competition is big. So, when there is strong competition it is very important to be innovative, but also successful in the same way. This is done by analysis and predicting the success of new products. For example, social data analytics can be used by fashion industry brands to see the reactions of their followers about new products. They can also use it to see which types of user groups their followers belong to, in order to make right predictions in the future. Fashion companies can use it to decide what is the best way to launch specific products. It is also useful to gather information whether their followers are students, mid-age or older people, which of two genders are reacting better and what to do to stimulate their better reaction. Another use case of social data analysis is discovering the influential entities inside a social network (people, places etc.) and discovering trends and trending topics

2.2 Commercial tools used for social data analytics

Some of the most used tools that have been used for social data analytics are Sprout social[1], Buzzsumo[2] and Google analytics[3].

[1] https://sproutsocial.com/

Sprout social can measure performance across Facebook, Twitter, Instagram and LinkedIn, all within a single platform. Having analytics at one place makes it easier to track and compare the company's efforts across multiple profiles and platforms. It is recommended for any brand that manages multiple social media profiles across multiple networks.

Buzzsumo is different than the other social media analytics tools on our list. Instead of analysing your brand's individual social media performance, Buzzsumo looks at how content from the company's website performs on social media. For instance, if one wants to see how many shares the latest blog post received on Facebook and Twitter, Buzzsumo can provide hat data. Buzzsumo not only shows the number of shares for each piece of content, but also which type of content performs best on each network based on length, type, publish date and more.

Google analytics is not technically a "social media analytics tool," it is one of the best ways to track social media campaigns and even help measure social return on investment. Most companies likely already have GA setup on their website to monitor and analyse your traffic.

The previous tools use statistical techniques for analysing the social network data. While they are able to analyse user reactions, they don't support the analysis of contents left by users, such as comments and posts. For this part it is useful to have some machine learning techniques for analysing text context and that is what we recognize and tries to create a complete API that includes all modules together.

2.3 Open source libraries and frameworks

Besides previously mentioned commercial tools, there are many open source solution like libraries, modules frameworks for sentiment analysis, clustering and natural language processing that can be used to build automated systems for specific tasks like for example Twitter opinion mining or Facebook user base clustering. Some of these tools were also used for our project.

NLTK[4] is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. It is usually used for preprocessing but it also includes out of the box algorithms for text classification.

[2] http://buzzsumo.com/
[3] https://analytics.google.com/
[4] http://www.nltk.org/

VADER[1] (Valence Aware Dictionary and Sentiment Reasoner) Sentiment Analysis is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media, and works well on texts from other domains.

Scikit-learn[5] is a machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including SVM, random forests, neural networks, Naive Bayes, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Sklearn can be a base for building custom-made machine learning solutions for analysis of social data. It supports all parts of creating machine learning model, from pre-processing and training to evaluating the results.

2.4 Research work

With the population of blogs and social networks, opinion mining and sentiment analysis became a field of interest for many researches. A broad overview of the existing work was presented in (Pang and Lee, 2008)[10]. In their survey, the authors describe existing techniques and approaches for an opinion-oriented information retrieval.

J. Read in (Read, 2005)[11] used emoticons such as ":-)" and ":- (" to form a training set for the sentiment classification. For this purpose, the author collected texts containing emoticons from Usenet newsgroups. The dataset was divided into "positive" (texts with happy emoticons) and "negative" (texts with sad or angry emoticons) samples. Emoticons- trained classifiers: SVM and Naive Bayes, were able to obtain up to 70% of an accuracy on the test set.

In (Go et al., 2009)[12], authors used Twitter to collect training data and then to perform a sentiment search. The approach is similar to (Read, 2005). The authors construct corpora by using emoticons to obtain "positive" and "negative" samples, and then use various classifiers. The best result was obtained by the Na¨ıve Bayes classifier with a mutual information measure for feature selection. The authors were able to obtain up to 81% of accuracy on their test set. However, the method showed a bad performance with three classes ("negative", "positive" and "neutral")

We also considered the problem of spam filtering using machine learning techniques, as it is useful to separate such data from the ones that have real information and are thus useful for analysis.

W.A. Awad and S.M. ELseuofi (2011) [4] give a brief overview of machine learning techniques and algorithms suitable for spam filtering. They compared Naive Bayes, SVM, k-nearest neighbors, neural

[5] http://scikit-learn.org/

networks and few more approaches and concluded that Naive Bayes has the overall best performance in terms of precision, recall and accuracy.

Paul Attewell, David B. Monaghan and Darren Kwon [6] give an overview of general data mining process. They explain all steps and the challenges after performing data mining methods, from pre-processing techniques, such as ways of handling missing values, one-hot encoding, to how to interpret the obtained outputs.

In the book "Pattern Recognition And Machine Learning" by Christopher M. Bishop [5] (Springer, 2006) we find more detailed explanations about the machine learning algorithms like K-means and Naive Bayes. There are also well-explained concepts about dimensionality reduction techniques like Principal Component Analysis. The author explains the mathematical background of the technique and also gives examples where this technique can be used. One of the examples was dimensionality reduction technique for data visualization purpose, which we use in our project.

Chapter 3

Machine Learning Spam Filter

Machine learning is a subfield from the broad field of artificial intelligence, this aims to make machines able to learn like human. Learning here means understanding, observing and representing information about some statistical phenomenon. In unsupervised learning one tries to uncover hidden regularities (clusters) or to detect anomalies in the data like spam messages or network intrusion. In spam filtering task, features could be the bag of words or the subject line analysis. Thus, the input to the classification task can be viewed as a two-dimensional matrix, whose axes are the messages and the features. Classification tasks are often divided into several sub-tasks. First, Data collection and representation are mostly problem- specific, second, feature selection and feature reduction attempt to reduce the dimensionality (i.e. the number of features) for the remaining steps of the task. Next step is the actual training of the classifier based on these features, from which the classifier builds a model of the training data that tries to generalize as much as possible in order for it to be useful for the samples that are not yet seen. Finally, the classification phase of the process finds the actual mapping between the message and whether it belongs to the certain class (in the case of spam filtering: spam or ham).

There are various algorithms and approaches to achieve this task, and in the following sections we are going to justify why we decided to use the Naïve Bayes approach for detecting whether some social content is spam or not. First let's review briefly some of the machine learning algorithms that are often used for this purpose.

3.1 Comparison of machine learning algorithms for spam filtering

In "Machine Learning Methods for Spam E-mail classification"[4] W.A. Awad and S.M. ELseuofi tested the performance of several algorithms in the task of spam classification. Corpora of spam and legitimate messages was compiled which contains total of 6000 messages of which 37% are spam. The messages were modelled as bag-of-words, where the top 100 most frequent words used in the whole corpora were chosen as features. The performance of the algorithms was presented in terms of spam recall, precision, and accuracy. Algorithms that were compared were: Naïve Bayes (**NB**), Support Vector Machine (**SVM**), K

– nearest neighbours (**KNN**), Neural Network (**NN**), Artificial Immune System (**AIS**) and Rough Set (**RS**).

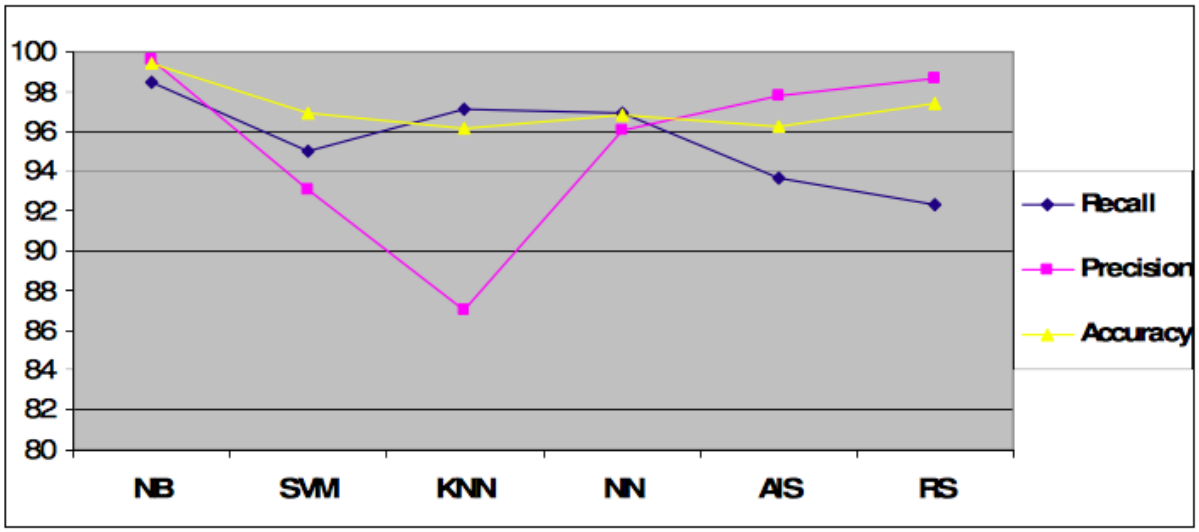

Figure 1: Comparison of machine learning algorithms for spam classification

In terms of accuracy we can find that the **Naïve Bayes** method is the most accurate while the **Artificial Immune System** and the **k- nearest neighbour** gives us approximately the same lower percentage. In terms of spam precision we can find that the **Naïve Bayes** method has the highest precision among the six algorithms while the **k-nearest neighbour** has the worst precision percentage and surprisingly the **Rough Sets** method has a very competitive percentage. Finally we can find that the **Rough Sets** recall has the less percentage among the six classifiers while the **Naïve Bayes** still has the highest performance.

Overall from the chart above we can see that Naïve Bayes has the best performances in terms of precision, recall and accuracy, which was a reason why we chose it as an approach for training and implementing social networks content spam filter.

In the next chapter, Naïve Bayes algorithm is described in detail because it is important to understand the mathematical background in order to justify why it works for the specific task of classifying spam content.

3.2 Naïve Bayes classifier method

Naive Bayes classifier is a linear classifier based on the popular Bayes' probability theorem, and it is known for creating simple yet well performing models, especially in the fields of document classification.

Naïve come from the assumption that the features of the dataset used for learning are mutually independent, which is very contradictory to the reality, since this assumption is rarely true, but surprisingly Naïve Bayes tends to perform very well under this unrealistic assumption.

Bayes' theorem forms the core of the whole concept of naive Bayes classification.[6]

$$a - posteriori\ probability = \frac{conditional\ probability * prior\ probability}{evidence}$$

The a-posteriori (or posterior) probability can be interpreted as: "what is the probability that the object belongs to the class *i* given its observed feature value. The general notation of posterior probability can be written as:

$$P(W_j \mid x_i) = \frac{P(x_i \mid W_j) * P(W_j)}{P(x_i)}$$

Where $\boldsymbol{x_i}$ is the feature vector of sample *i*, and $\boldsymbol{W_j}$ is the notation for the class *j.* $\boldsymbol{P(x_i \mid W_j)}$ is the probability of observing feature, given that the sample belongs to the class $\boldsymbol{W_j}$.

One assumption that Bayes classifiers make is that the samples are *i.i.d.* The abbreviation *i.i.d.* stands for "independent and identically distributed" and describes random variables that are independent of one another and are drawn from a similar probability distribution. Independence means that the probability of one observation does not affect the probability of another observation (e.g., time series and network graphs are not independent). One popular example of *i.i.d.*variables is the classic coin tossing: The first coin flip does not affect the outcome of a second coin flip and so forth. Given a fair coin, the probability of the coin landing on "heads" is always 0.5 no matter of how often the coin if flipped.

An additional assumption of naive Bayes classifiers is the *conditional independence* of features. Under this *naive* assumption, the *class-conditional probabilities* or (*likelihoods*) of the samples can be directly estimated from the training data instead of evaluating all possibilities of $\mathbf{x}$. Thus, given a d-dimensional feature vector $\mathbf{x}$, the class conditional probability can be calculated as follows:

$$P(x \mid W_j) = P(x_1 \mid W_j) * P(x_2 \mid W_j) * \ldots * P(x_d \mid W_j) = \prod_{k=1}^{d} P(x_k \mid W_j)$$

[6] https://arxiv.org/abs/1410.5329/

Here, $P(x \mid W_j) =$ simply means: "How likely is it to observe this particular pattern **X** given that it belongs to class W_j. The "individual" likelihoods for every feature in the feature vector can be estimated via the maximum-likelihood estimate, which is simply a frequency in the case of categorical data:

$$P(x_i \mid W_j) = \frac{N_{x_i w_J}}{N_{w_j}}$$

Where the denominator is the total count of samples of class W_j, and the numerator represents the number of times a feature x_i appears in samples of class W_j.

To illustrate this concept with an example, let us assume that we have a collection of documents where some documents are *spam* messages. Now, we want to calculate the class-conditional probability for a new message "Hello World" given that it is spam. Here, the pattern consists of two features: "hello" and "world," and the class-conditional probability is the product of the probability of encountering "hello" given that the message is spam with the probability of encountering "world" given that the message is spam.

$$P(x = [hello, world] \mid spam) = P(hello \mid spam) * P(world \mid spam)$$

Prior probability (or just prior) is introduced that can be interpreted as the prior belief or a-priori knowledge. In the context of pattern classification, the prior probabilities are also called class priors, which describe "the general probability of encountering a particular class." In the case of spam classification, the priors could be formulated as:

$$P(spam) = \frac{\#\ of\ spam\ messages\ in\ the\ dataset}{\#\ of\ all\ messages\ in\ the\ dataset}$$

$$P(ham) = 1 - P(spam)$$

After defining the class-conditional probability and prior probability, there is only one term missing in order to compute posterior probability, that is the evidence. The evidence $P(x)$ can be understood as the probability of encountering a particular pattern **x** independent from the class label. Although the evidence term is required to accurately calculate the posterior probabilities, it can be removed since it is merely a scaling factor.

The rule for deciding whether to classify a sample with a certain class $P(W_j \mid x_i) =$ is simply:

Classify sample x_i as W_1 if $P(W_1 \mid x_i) > P(W_2 \mid x_i)$ else classify the sample as W_2, in case where we have two classes.

Now that we have a mathematical background about the classification algorithm that is going to be used for implementing the spam filter, we can proceed with a detailed explanation of the dataset used for training the classifier, pre-processing and feature extraction, as well as the concrete implementation of Naïve Bayes spam filter for social data.

3.3 Naïve Bayes implementation of spam filter for social data

3.3.1 Dataset

The dataset we used for training the Naïve Bayes spam filter is YouTube Spam collection from the UCI Machine Learning Repository[7]. The dataset consists of YouTube comments that have been labelled as spam or ham (not spam). The spam filter will be used to classify social content, thus for comments on social media it resulted natural to use this dataset, since the contents of the comments on videos and social networks are similar.

The collection consists of 1956 rows of comments on various YouTube videos. Among those, exactly 1005 comments were labelled as spam. This means the dataset is very balanced and easier to train the classifier on. In the ideal case, the dataset used to train the classifier should be content of a specific social network that the spam filter is going to be used on, and in fact the classifier can be re-trained on this other collection if one wishes to.

The rows in the dataset have five attributes: COMMENT_ID, AUTHOR, DATE, CONTENT, TAG
and the label associated with the class. In our case the only useful attribute was the actual content of the comment, so the first step in pre-processing this dataset was to drop all columns except the CONTENT column. After this, we are left with a 1956x2 matrix where the first column is the content and the second is the label (1-spam, 0-ham). Next step is to perform some feature extraction and selection on the dataset, meaning that it is necessary to extract the useful features on which the spam filter will be trained.

[7] https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection/

3.3.2 Bag of Words model

Important sub-tasks in pattern classification are *feature extraction* and *selection*; the three main criteria of good features are listed below:

- *Salient*. The features are important and meaningful with respect to the problem domain.
- *Invariant*. The features are insusceptible to distortion, scaling, orientation, etc.
- *Discriminatory.* The selected features bear enough information to distinguish well between patterns when used to train the classifier.

Prior to fitting the model and using machine learning algorithms for training, we need to think about how to best represent a text content as a feature vector. A commonly used model in Natural Language Processing is the so-called bag of words model. The idea behind this model really is as simple as it sounds. First comes the creation of the vocabulary — the collection of all different words that occur in the training set and each word is associated with a count of how it occurs. This vocabulary can be understood as a set of non-redundant items where the order doesn't matter. Let D1 and D2 be two comments in a training dataset:

- D_1: "Hey, http://believemefilm.com"
- D_2: "Hey, Katy Perry reminds me of a tiger"

Based on these two comments, the vocabulary could be written as:

V = {**hey**: 2, **katy**: 1, **perry**: 1, **reminds**: 1, **me**: 1, **of**: 1, **a**: 1, **tiger**: 1, **http://believemefilm.com**: 1}

The vocabulary can then be used to construct the d-dimensional feature vectors for the individual documents where the dimensionality is equal to the number of different words in the vocabulary (d=|V|). This process is called *vectorization*.

Table 1: Bag of words representation of two sample documents D1 and D2

	hey	**katy**	**perry**	**reminds**	**me**	**of**	**a**	**tiger**	**http://believemefilm.com**
x_{d1}	1	0	0	0	0	0	0	0	1
x_{d2}	1	1	1	1	1	1	1	1	0

This is done for every row on the CONTENT column. Since in our dataset number of different words in the vocabulary is huge, we reduced this feature vector to the top 300 most frequent words in the dataset and by doing this the classifier focuses only on the important features not some words that occur once in 1956 comments.

Before the feature extraction and transformation to bag of words model, there are some other steps applied to pre-process the textual data so that we could get the best possible features. These steps are: tokenization, stop words elimination, stemming and lemmatization. The tool used for all the processing of natural language and text is NLTK library for Python3.

3.3.3 Tokenization

Tokenization describes the general process of breaking down a text corpus into individual elements that serve as input for various natural language processing algorithms. Usually, tokenization is accompanied by other optional processing steps, such as the removal of stop words and punctuation characters, stemming or lemmatizing, and the construction of n-grams. Below is a simple example of tokenization.

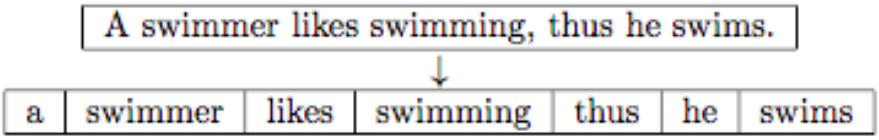

Figure 2: Sentence Tokenization

3.3.4 Stop Words

Stop words are words that are particularly common in a text corpus and thus considered as rather un-informative (e.g., words such as so, and, or, the, …"). One approach to stop word removal is to search against a language-specific stop word dictionary. An alternative approach is to create a stop list by sorting all words in the entire text corpus by frequency. The stop list — after conversion into a set of non-redundant words — is then used to remove all those words from the input documents that are ranked among the top n words in this stop list.

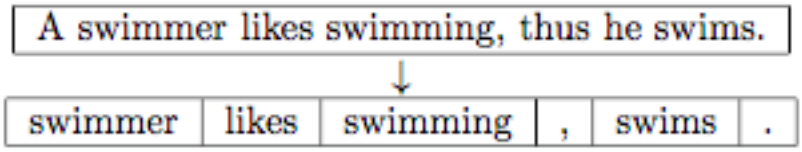

Figure 3: Stop words removal

The stop words that we considered are the word defined in the *stopwords* corpora of NLTK library. Some of them are:

ıdɛ ▲	Type	Size	Value
0	str	1	i
1	str	1	me
2	str	1	my
3	str	1	myself
4	str	1	we
5	str	1	our
6	str	1	ours
7	str	1	ourselves
8	str	1	you
9	str	1	your

Figure 4: NLTK stop words

Also, punctuation marks were appended to the list of the stop word, such as **[‘[‘, ‘]’, ‘.’, ‘,’, ‘:’, ‘;’, ‘(‘, ‘)’]** since they don’t carry any valuable information for the training the classifier.

3.3.5 Stemming and Lemmatization

Stemming describes the process of transforming a word into its root form. The original stemming algorithm was developed by Martin F. Porter in 1979 and is hence known as Porter stemmer. Stemming can create non-real words, such as “thu” in the example below.

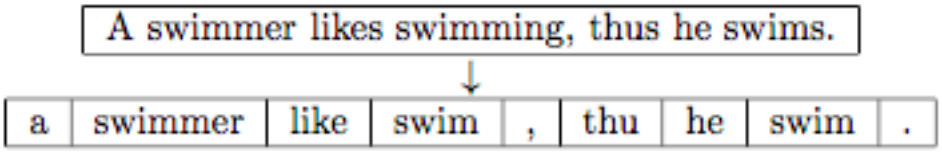

Figure 5: Stemming

In contrast to stemming, lemmatization aims to obtain the canonical (grammatically correct) forms of the words, the so-called lemmas. Lemmatization is computationally more difficult and expensive than stemming.

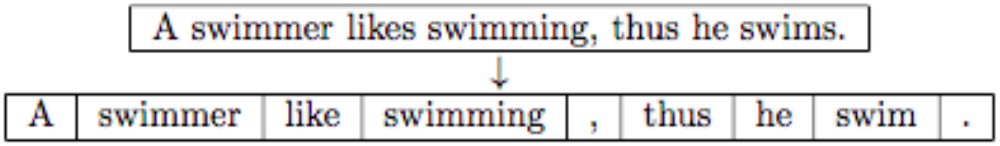

Figure 6: Lemmatization

Given the information above we choose to only apply lemmatization in the pre-processing of features, not stemming. The lemmatizer used is the WordNetLemmatizer from NLTK library.

3.3.6 Training the Naïve Bayes classifier

Now that each row of the training dataset was first tokenized then lemmatized and all the stop words and punctuation marks were remove, the collection is almost ready for training the spam filter classifier. As was said earlier, since the vocabulary in this case is huge, we only chose to use the first 300 most frequent words used in the dataset. Some of these words are:

ıdɛ ▲	Type	Size	Value
0	str	1	huh
1	str	1	anyway
2	str	1	check
3	str	1	tube
4	str	1	channel
5	str	1	kobyoshi02
6	str	1	hey
7	str	1	guy
8	str	1	new
9	str	1	first

Figure 7: Most frequent words from the dataset

Every row is transformed into the bag of words model as was described earlier and the dataset is ready for training. The library used for training the Naïve Bayes model is sklearn for Python, which used the Gaussian Naïve Bayes and works very similarly to the method described earlier, except it assumes Gaussian distributions of samples. After the training is done, the classifier is ready to use. In order to validate the quality of the generated model, 75:25 holdout test dataset was used (original dataset was split, and 25% of it was left for evaluation). It is very important to use a separate set of data for evaluation of the model, in order to test how well the classifier is generalizing, meaning how well it will perform on never before seen data. If we used the same training dataset for evaluation also, then the model might be overfitting, and it would have excellent results on this dataset, but in reality, on new data it would perform very poorly.

Metrics used for evaluation of the classifier are all based on the confusion matrix. A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa).

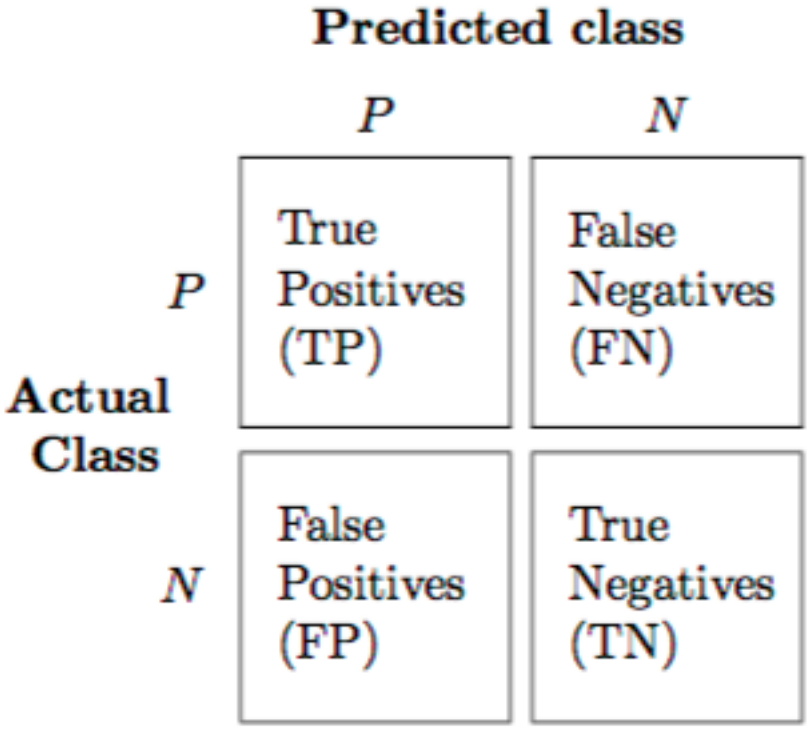

Figure 8: Confusion Matrix

The output of the evaluation of spam filter and the confusion matrix in this case is:

Table 2: Spam filter confusion matrix

	Positive	Negative
Positive	203	0
Negative	30	93

There are several scores that are relevant for evaluation of the classifier, which are calculated based on the confusion matrix. These scores are:

Accuracy - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model. For our model we have got 0.907, which means our model is approximately 90% accurate.

Accuracy = TP+TN/TP+FP+FN+TN

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate. We have got 1.0 precision, since there are no false positives, which means that precision might not be the best metrics to evaluate the model. But having no FP might be good for spam filter, since it's better to classify something as not spam, then to lose that information, in the case of social media content.

Precision = TP/TP+FP

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. We have got recall of 0.756 which is good for this model as it's above 0.5.

Recall = TP/TP+FN

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall. In our case, F1 score is 0.861.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Chapter 4

Automated sentiment analysis of social network content

4.1 Social network content

Data that is most useful for sentiment analysis on social networks is user-generated content, which includes comments, tweets, posts etc. In the case of our problem we focused on calculating the sentiment from comments on posts in order to automate feedback from the user base on the published post. The output of this module is the combined sentiment from all comments on a certain post, excluding spam comments (described in chapter 4). The sentiment is calculated from both text and emoji data on each comment and then it is combined as a single result in the range from -1 to 1, where the negative values denote negative sentiment/opinion, positive values positive sentiment/opinion and 0 means that the analysed content has neutral sentiment.

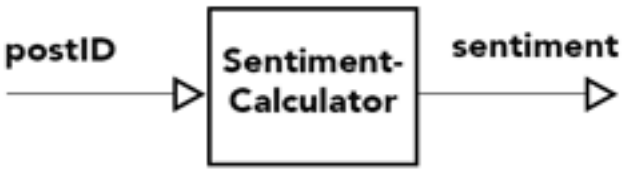

Figure 9: Overview of the Sentiment module

4.1.1 Database

The contents of the social media are stored in a database that has tables including attributes such as comments, posts, social media channels, users etc.

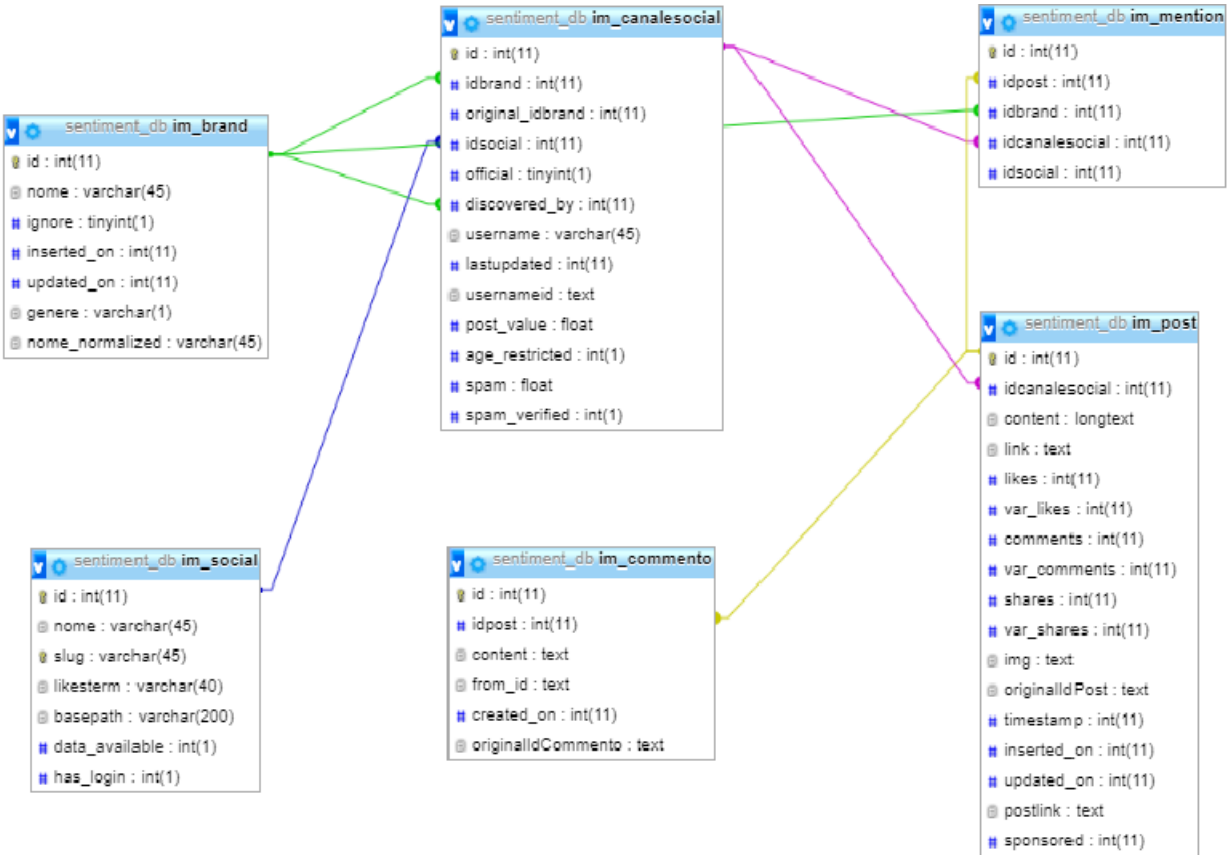

Figure 10: Social Media Channels Database

The table we are interested in is comments, which holds content of the comments (including text and emoji encoded in extended utf-8), user that posted it, corresponding postId and other meta-data. The input into the SentimentCalculator module in this case is the postId with which we can fetch all the corresponding comments.

4.2 SentimentCalculator

SentimentCalculator is used to calculate sentiment results given the text, or array of textual data, including text and emojis. In our case the input of SentimentCalculator will be an array of comments from a certain post and the result is the combined sentiment score in the range -1 to 1.

SentimentCalculator is developed in Python 3, using multiple libraries and packages, including google-translate for translation (since the content can be in any language) and VaderSentiment for sentiment calculation. The workflow of SentimentCalculator is the following:

- Take the array of text (emojis included) as input
- For each item (in this case social network comment) check if it's spam (using machine learn spam filter (chapter 4)
- If it is not spam, split the comment into two part – text and emojis
- Calculate sentiment for text part (using VaderSentiment)
- Calculate emoji sentiment (using custom emoji sentiment calculator)
- Combine both score into a single one (using some metrics)
- Combine scores for each comment into a single result (per post)

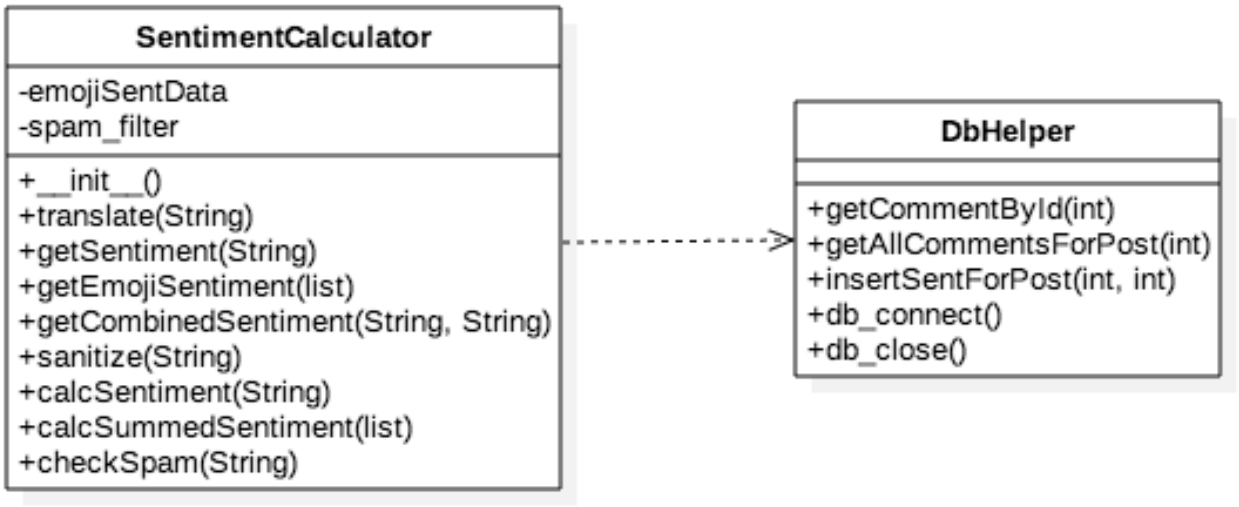

Figure 11: SentimentCalculator class diagram

At the end of the computation the result is the total sentiment of all the comments for a given post (or, whatever array of text and emoji data was provided as input), which can be used for further analysis and different fields. It is important to note that SentimentCalculator can be used as a generic module for different kinds of content. It is enough that the input is an array of text (and possibly emoji) data, which can come from different sources (doesn't have to be social media content).

4.2.1 Text Sentiment

Text sentiment determines if a text is positive, negative, or neutral, and to what degree (in the range -1 to 1). In our scope text sentiment refers only to the textual part of the social media comment, and it is calculated using VaderSentiment [1]. VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically tuned to sentiments expressed in social media. It is fully open-sourced under the MIT licence.

The compound score is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive). This is the most useful metric if you want a single uni-dimensional measure of sentiment for a given sentence. We call it a "normalized, weighted composite score".

It is also useful for researchers who would like to set standardized thresholds for classifying sentences as either positive, neutral, or negative. Typical threshold values are:

- **positive sentiment**: compound score >= 0.5
- **neutral sentiment**: (compound score > -0.5) and (compound score < 0.5)
- **negative sentiment**: compound score <= -0.5

The pos, neu, and neg scores are ratios for proportions of text that fall in each category (so these should all add up to be 1 - or close to it with float operation).

The condition that needs to be satisfied in order to use VaderSentiment is that the textual data needs to be in English. Thus, as a preprocessing step, each comment is translated using Google translate (googletrans Python package).

4.2.2 Emoji Sentiment

Emoji sentiment is calculated using emoji sentiment data stored in json format based on the work of of Kralj Novak, Petra; Smailović, Jasmina; Sluban, Borut and Mozetič, Igor [2]. They engaged 83 human annotators to label over 1.6 million tweets in 13 european languages by sentiment polarity (negative, neutral or positive). Example of an emoji sentiment datum:


```
[
  ...
  {
    // original properties:
    "sequence": "1F602",
    "occurrences": 14622,
    "negative": 3614,
    "neutral": 4163,
    "positive": 6845,
    // derived properties:
    "pNegative": 0.24717948717948718,
    "pNeutral": 0.2847179487179487,
    "pPositive": 0.4681025641025641,
    "score": 0.22092307692307694,
    "sem": 0.006751317877016391
  },
  ...
]
```


Figure 12: Emoji sentiment datum

Here is the explanation of the properties of an emoji sentiment datum:

- `sequence` (original): normalized code point sequence (sequence without any variation selector or modifier applied) e.g. `1F602`; it is used for mapping the sentiment datum to a specific (emoji) unicode character or for connecting it with further meta data (e.g. unicode-emoji-data, unicode-emoji-annotations or emoji-datasource)

- `occurrences` (original): absolute number of occurrences of the (emoji) unicode character in tweets

- `negative` (original): absolute number of occurrences of the (emoji) unicode character in tweets labeled negative

- `neutral` (original): absolute number of occurrences of the (emoji) unicode character in tweets labeled neutral

- `positive` (original): absolute number of occurrences of the (emoji) unicode character in tweets labeled positive

- `pNegative` (derived): relative negativity component of the sentiment distribution for those tweets associated with the (emoji) unicode character, ranging from `0` to `1`

- `pNeutral` (derived): relative neutrality component of the sentiment distribution for those tweets associated with the (emoji) unicode character, ranging from `0` to `1`

- `pPositive` (derived): relative positivity component of the sentiment distribution for those tweets associated with the (emoji) unicode character, ranging from `0` to `1`

- `score` (derived): resulting sentiment score of the (emoji) unicode character, ranging from `-1` to `+1`, calculated as the mean of the discrete sentiment distribution of `negative` (`-1`), `neutral` (`0`) and `positive` (`+1`)

- `sem` (derived): precalculated Standard Error Mean for further deriving the confidence interval, e.g. for 95%: `[score – 1.96 * sem, score + 1.96 * sem]`

The sum of `negative`, `neutral` and `positive` is `occurrences`.
The sum of `pNegative`, `pNeutral` and `pPositive` is `1`.

We extended this by neutralizing all sentiments for emojis that had very few occurrences, since the score is unreliable, and it's better to assign a neutral sentiment to the given emoji that use this score that can lead to false conclusions.

The input to the emoji sentiment calculation method is an array of all the emojis extracted from the comment, duplicates included. The algorithm for emoji sentiment calculation from SentimentCalculator is the following:

- If there are no emojis (i.e if the array has length 0) then the emoji sentiment is neutral (0)
- For each emoji get sentiment from the json file mentioned earlier and sum it to the previous sentiment
- Every three occurrences of the same emoji increase it's importance by a small amount (this is mentioned in "Spice up your Chat: The intentions and Sentiment Effects of using Emojis"[3])
- At the end, normalize the summed emoji sentiment in order to be in the range -1 to 1

The output of the emoji sentiment calculation is the same as the output from the text sentiment calculation, and it is used to enhance the total combined sentiment and to take into the account also the emoji content of the comment.

4.2.3 Combined Sentiment

Combined sentiment (i.e., total sentiment) is produced putting together text sentiment (3.2.1) and emoji sentiment (3.2.2). The output will again be normalized in the range -1 to 1. The formula for combining sentiments that was used is the following:

$$combined_{sentiment} = \frac{(text_{sentiment} + reg_{emoji} * emoji_{sentiment})}{2}$$

This is the formula for calculating *combined sentiment* if *reg* is different from*.* 0*. reg* is a regularization term for importance of emoji sentiment. If *reg* is 0 only text sentiment combined sentiment is equal to text sentiment.

4.3 Usage and results

SentimentCalculator can be used as a separate independent module, or in our case in correspondence with db_helper module which fetches the data from the comments table and forwarding it to the SentimentCalculator. SentimentCalculator can be used to calculate sentiment from a single comment by calling ***SentimentCalculator****.calcSentiment(comment)* or calculate the summed sentiment from all the comments of the given post by calling ***SentimentCalculator****.calcSummedSentiment(listComments),* where the list of comments is fethed from the database by the db_helper for a given postId. The results are then written back into the sentiment table using the db_helper. A more detailed description of the architecture of this module will be illustrated in Chapter 6.

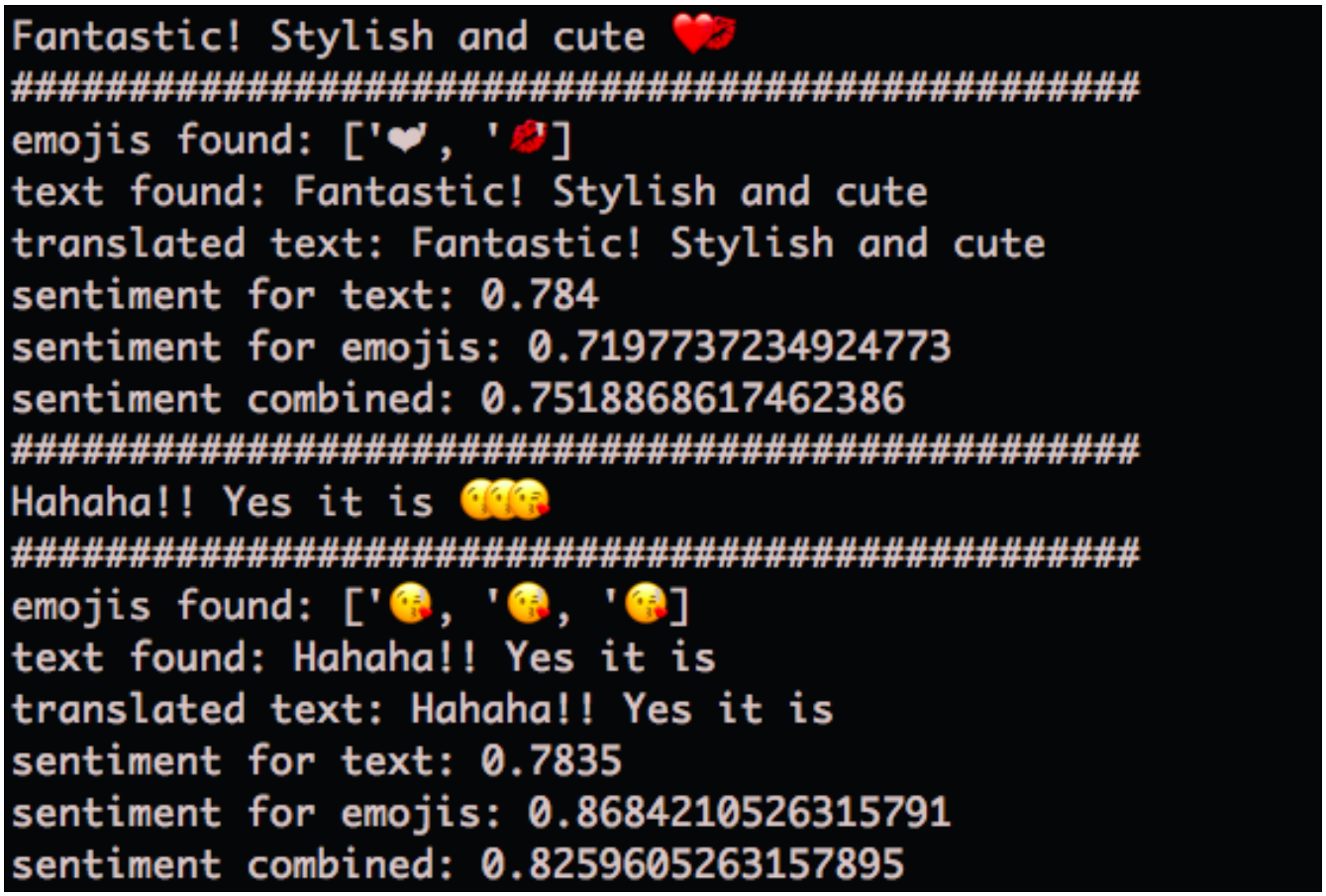

Figure 13: SentimentCalculator output

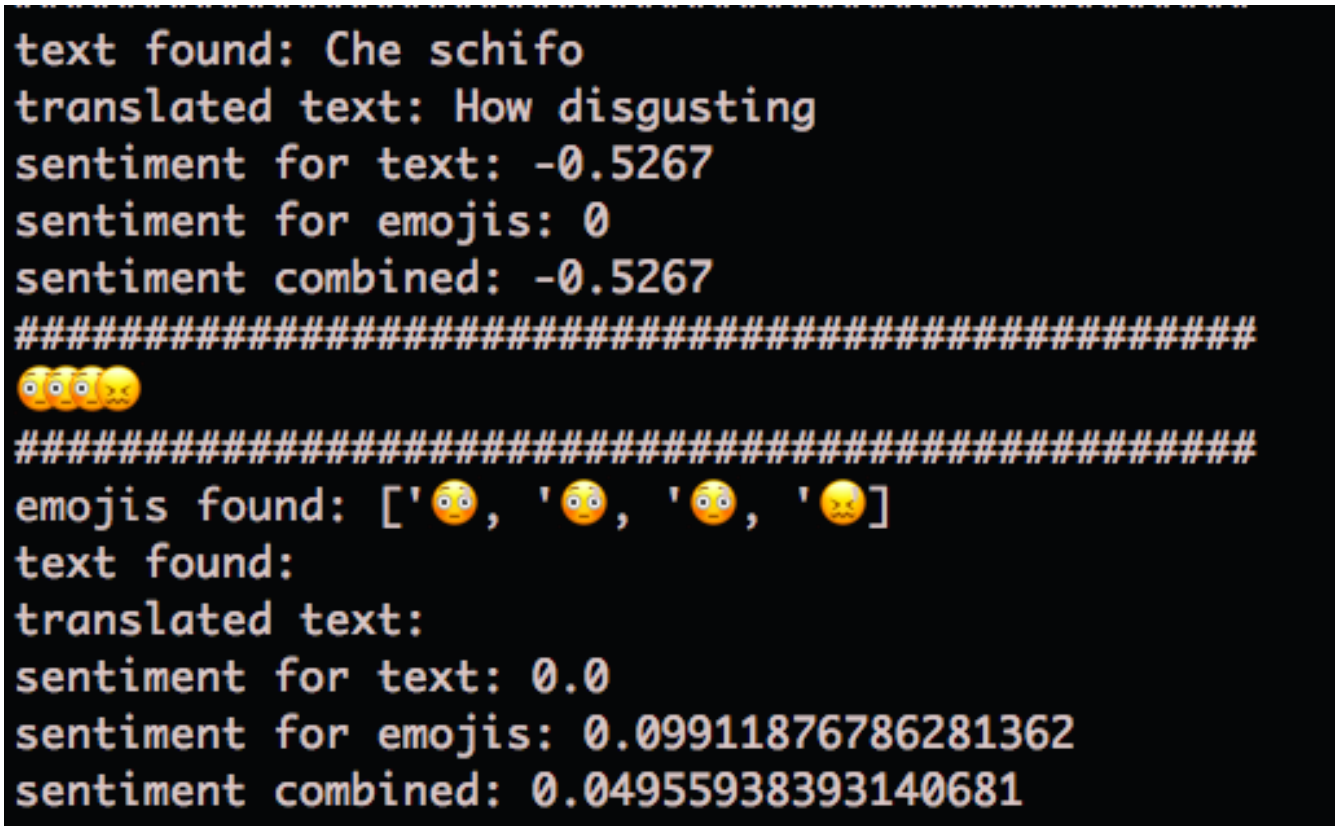

Figure 14: SentimentCalculator output example for negative comments

There is also a batched sentiment calculation, which calculates sentiment for all the post in the database. It iterates over all the posts, fetches all the comments for that post and calls the same ***SentimentCalculator****.calcSummedSentiment(listComments)* as before.

Chapter 5

Unsupervised approach to Social Data Analysis

Social data also includes data about the users like their gender, city, description, date of birth. These data can be used to detect similarities between the users. Similar users can be identified in the same group, so that it becomes easier to target their preferences within marketing campaigns. So, the main goal of this module is to find those groups of users and to visualize their characteristics.

5.1 Fetching data from social networks

To be able to analyse user data, first we need to gather those data. User data are present on social networks, but to gather them we need to have right permissions. For the purpose of this project we were provided a dataset that included comments, posts, social channels, brands and desired social networks with fallowing relationships:

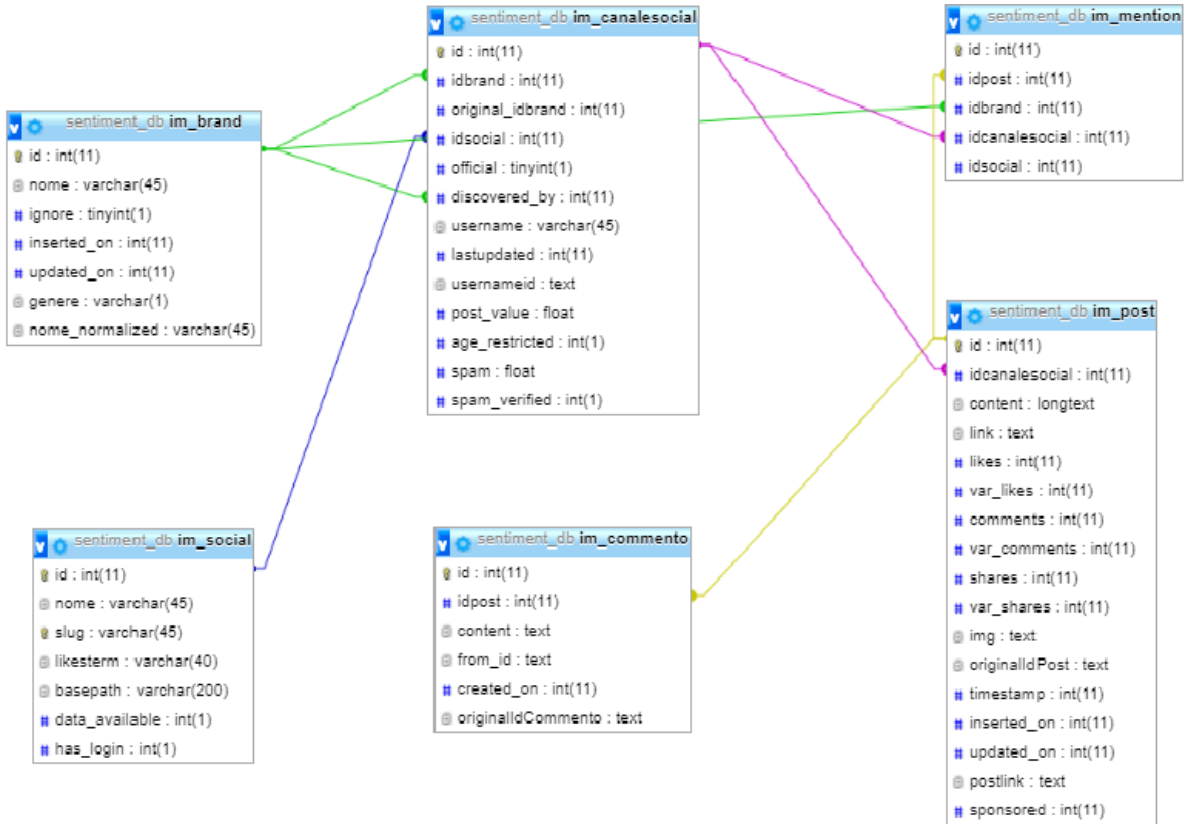

Figure 15: Sentiment_db

So, from the logic schema of the data we see that for every post from *im_post* table we can fetch all comments that are located in table *im_commento*. Chapter 2 explained how to determine summarised sentiment of all users about a post. Now, to go further and characterize the user groups that posted comments, we are interested in the characteristics and behaviour of each group. To get user data we need to use the API each social channel we are interested in. We will explain how Facebook API is used for this purpose and for the rest the procedure is very similar.

Facebook Graph API requests user authorization access before applying search. User connects to Facebook API by sending access token and to get it the developer need to register at *developers.facebook.com* and to register the application that will use the search. After successful authorization then the developer can do the search.

Id parameter is target user id and the other is list of user attributes that we are interested in. The problems we met are related to permissions and user privacy options, since not all users shared all desired attributes. So, our dataset is not full, and we used some pre-processing techniques to overcome this problem. This is explained in the next part of this chapter. The returned result is a JSON formatted string. All the data fetched are stored into *user_social* table.


```
{
   "birthday": "01/01/2010",
   "about": "Fashion, Trends, Beauty Style, Personal Shopper and Stylist.",
   "email": "info@driferreira.com",
   "city": "San Diego",
   "country": "United States",
   "category": "Personal Assistant",
   'description": "Dri Ferreira is a Stylist and Personal Shopper for the Fashionable Elite for... "
}
```


Figure 16: FB API response

5.2 Pre-processing

Getting the target data is one thing, but getting the information behind that data is something else. Raw data usually need some modifications, transformations or, in one word data, pre-processing.

In every project related to data science and machine learning, the pre-processing step takes almost 60% of time.

5.2.1 Exploratory data analysis

First thing that is usually done in pre-processing is exploratory data analysis. We are inspecting features (columns) of the dataset. We are inspecting basic statistic, types, visualize distributions,

correlations, and other interesting measures. In the next picture there is an example from the data of a user who commented on a specific post:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 599 entries, 0 to 598
Data columns (total 7 columns):
birthday        98 non-null object
 about          565 non-null object
 emails         314 non-null object
 city           265 non-null object
 country        248 non-null object
 category       599 non-null object
 description    355 non-null object
dtypes: object(7)
memory usage: 32.8+ KB
```

Figure 17: User dataset info

Also, data types can be numeric, categorical or string variables. Here we must pay great attention to string variables, since they are unstructured and enriched with hashtags, links and emojis. Also, content is generated on many different languages, so we need to translate it in order to have useful information. We can see examples on next dataset.

	birthday	about	emails	city	country	category	description
0	None	Let Us Make It Happen	None	None	None	Company	None
1	None	Authentic, Irriverent, Unpredictable, Passiona...	rome_sales@hardrock.com	Rome	Italy	American Restaurant	None
2	None	All About Top Trend Share	None	None	None	Website	None
3	04/06/2016	Una pizza di alta qualitÃ , con ingredienti DO...	info@thursdaypizza.com	Milan	Italy	Pizza Place	Thursday Pizza offre una pizza di qualitÃ , co...
4	None	SpÃ©cialiste de la prÃ©paration esthÃ©tique au...	contact@brillance-auto.fr	Paimpol	France	Cleaning Service	Brillance Auto propose :\n\n- Nettoyage intern...

Figure 18: User dataset table

5.2.2 Missing data

Datasets that are collected directly from users usually are not 100% full. We have to handle missing data in one of many ways following certain policy. For example, we chose to discard rows that have more than two missing fields. For the rest of the rows we fill missing fields using policy depending on the type of the field. For features that are:

- Categorical – we fill the missing value with most frequent category in specified feature
- Numerical – we selected median value since it is more resistant to outliers than median

In our dataset, all the variables was categorical, and we choose to fill-in missing data following distribution of values.

5.2.3 String data

String variables need to be handled in special way. We decided to transform them in categorical variables by fallowing next steps.

First, we need to have language consistency, so we used *Google Translator* to translate text to English. After that, we tokenize string by removing punctuations and stop words. Now we have a list of tokens instead of the string. Next, we create dictionary, where we count occurrence of each token and select top 10 since they take most of the variance of the dataset. Finally, we go again through all rows, calculate similarity between the tokens and categories, and select most similar categories. In that way we achieve a categorization of this string feature. We did this categorization on about and description features.

The birthday feature, which was also stored as a string, was split into three new integer features: day, month and year.

5.2.4 Categorical data

Categorical data are variables that often contain label values rather than numerical values. The number of possible values are limited to fixed set.

Some examples include:

- A “gender” variable with the values: “male” and “female “.
- A “vehicle” variable with the values: “car “, “motorcycle” and “truck “.
- A “language” variable with the values: “Italian”, “English” and “Serbian “.

Categorical data are perfect choice for some type of algorithms like decision trees. Although, there are also some algorithms that cannot work on label data directly, they require input and output variables to be numeric. This means that categorical data needs to be converted to a numeric form. Method for doing transformation is also used in this project has two steps:

1) *Integer encoding step* where each category is assigned an integer value. For example, “car” is one, “motorcycle” is two and “truck” is three.
2) *One-hot encoding* is necessary for categorical variables where no such relationship exists. One-hot encoding is applied to integer representation of categorical variable. This is where integer encoded variable is removed, and binary variable is added for each unique integer value. One

binary variable is redundant and thus it can be removed. Final binary variables are called *dummy variables*.

5.3 Unsupervised learning techniques

Unsupervised machine learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data. Classification or categorization problems are not included in this observation. Since the data in those problems are not label it is not possible to measure accuracy of the model output by the relevant algorithm, which is one difference between supervised and unsupervised machine learning algorithms.

5.3.1 Hierarchical clustering

Hierarchical clustering is one possible type of clustering. It can be agglomerative or divisive. Agglomerative clustering starts with all individual points as separate clusters and then starts merging them by similarity, which means in each step we merge most similar clusters until we get one cluster. Divisive clustering goes in other way. It starts with one cluster and at each step it separates into more similar clusters until it reaches number of clusters as there are data points. With this method we use dendrogram to visualise clustering and choose the best number of clusters. Advantage of this method is that we don't need to know number of clusters in advance and the disadvantage is the complexity of the algorithm which is time exhausting for big datasets.

5.3.2 K-Means

K-means clustering is another type of clustering that we applied in our project. Before we use it, we need to specify the number of clusters that we expect to have. At the beginning, random centroids whose number we specified are initialized and other data points are assigned to a cluster which is the closest centroid. The algorithm is specified in the following figure:

K-MEANS ($\mathbf{D}, k, \epsilon$):

$t = 0$

Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$

repeat

  $t \leftarrow t + 1$

  $C_j \leftarrow \emptyset$ for all $j = 1, \cdots, k$

  `// Cluster Assignment Step`

  foreach $\mathbf{x}_j \in \mathbf{D}$ **do**

    $j^* \leftarrow \arg\min_i \left\{ \left\| \mathbf{x}_j - \boldsymbol{\mu}_i^t \right\|^2 \right\}$ `// Assign` $\mathbf{x}_j$ `to closest centroid`

    $C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$

  `// Centroid Update Step`

  foreach $i = 1$ *to* k **do**

    $\boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$

until $\sum_{i=1}^{k} \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\|^2 \leq \epsilon$

Figure 19: K-Means algorithm

An advantage of this algorithm is that it is computationally fast and it also produces tighter clusters, especially if the clusters are globular. A disadvantage is that we need to know number of clusters in advance and it is not good for clusters with non-globular shape or with variating density. This algorithm had the best performance on our data, so we chose it for our dataset. The number of clusters was determined using the Elbow analysis method, explained in the following.

5.4 Elbow analysis

As we mentioned before, we have chosen K-means clustering technique to inspect our dataset. It had the best performance, but we still need to find the best number of clusters. So, to be able to do that, we need to measure performance for each trial number. We used two measures:

- Within-cluster sum of squares (WSS):

$$WSS(C) = \sum_{i=1}^{k} \sum_{x_j \epsilon C_i} \left| \left| x_j - \mu_i \right| \right|^2$$

where μ_i is the centroid of cluster C_i (in case of Euclidean spaces)

- Between-cluster sum of squares (BSS):

$$BSS(C) = \sum_{i=1}^{k} |C_i| * ||\mu - \mu_i||^2$$

where μ is the centroid of the whole dataset

For each number of clusters, we are making a WSS/BSS trade-off. We can plot this and use elbow method to determine optimal number of clusters. We can see that on next figure:

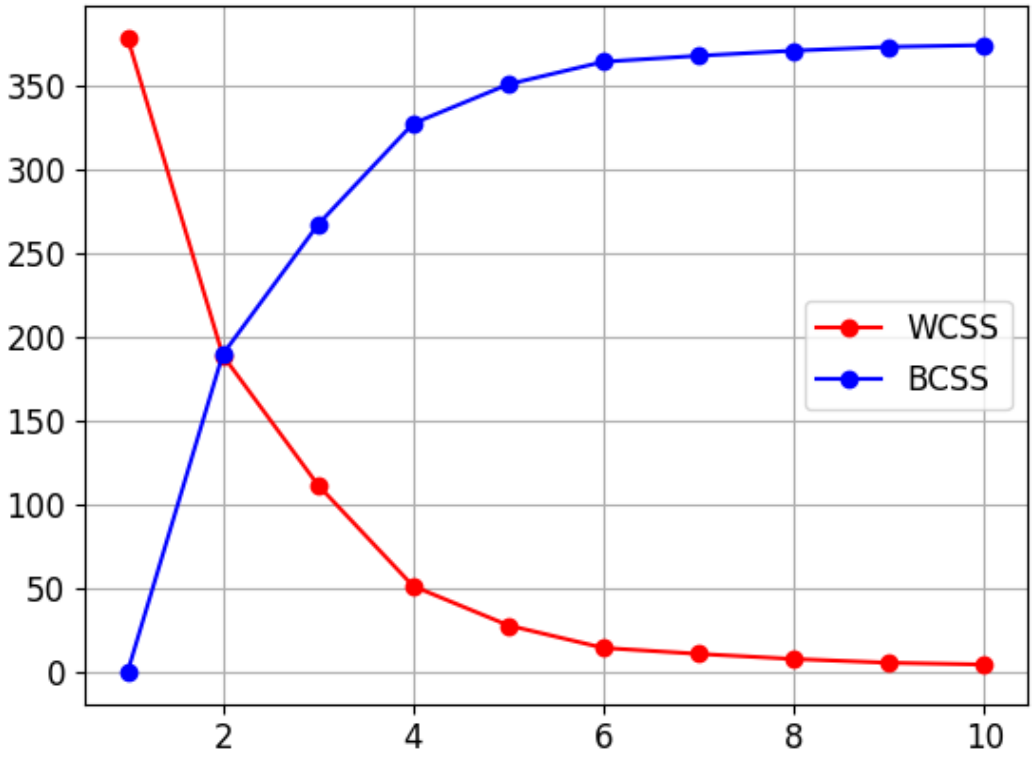

Figure 20: Knee for K-means clustering

From the graph we need to find one point after which WSS is stabilized. In our case, the x-axis represents number of clusters and we can see that after four clusters WSS is not changing much, i.e., it is stabilized. So, in our case the optimal number of clusters is four.

5.5 Visualization of clusters

Once we have done clustering on the dataset and we get the result, we need to understand the meaning of it, i.e. the characteristics of each cluster. In our project, all the variables are categorical, so we decided to visualize the distribution of values for each feature labelled by specific cluster.

5.5.1 Principal Component Analysis

Principal component analysis (PCA) is a technique used for dimensionality reduction, data compression, feature extraction and data visualization. PCA represents the orthogonal projection on a

lower-dimensional linear space, which is also known as principal subspace, in such a way that the variance of the projected data is maximized.

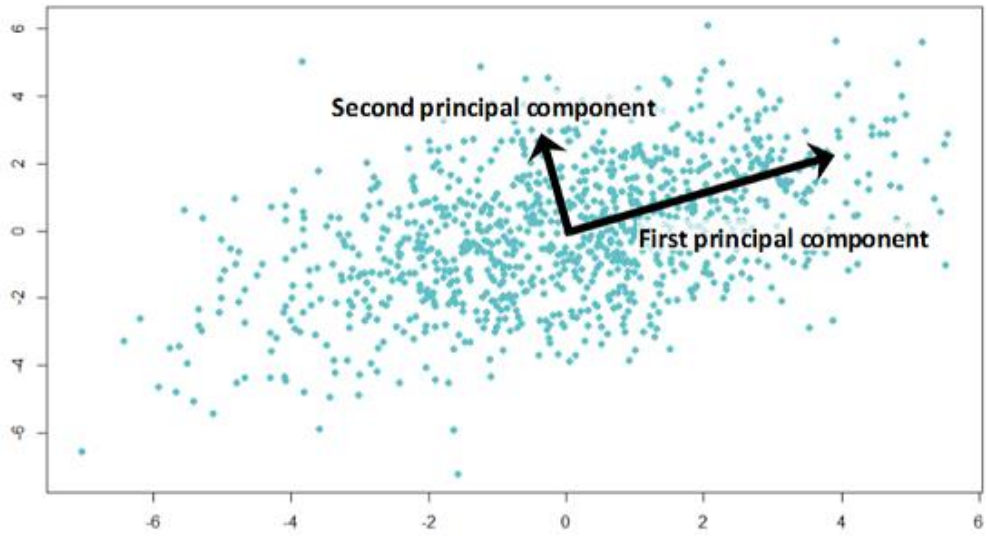

Figure 21: Example of PCA

In our project we used PCA for dimensionality reduction of multidimensional user dataset to be able to visualize clustering results. Clustering of dataset is shown in the next figure, where the dataset is clustered into 4 distinct groups.

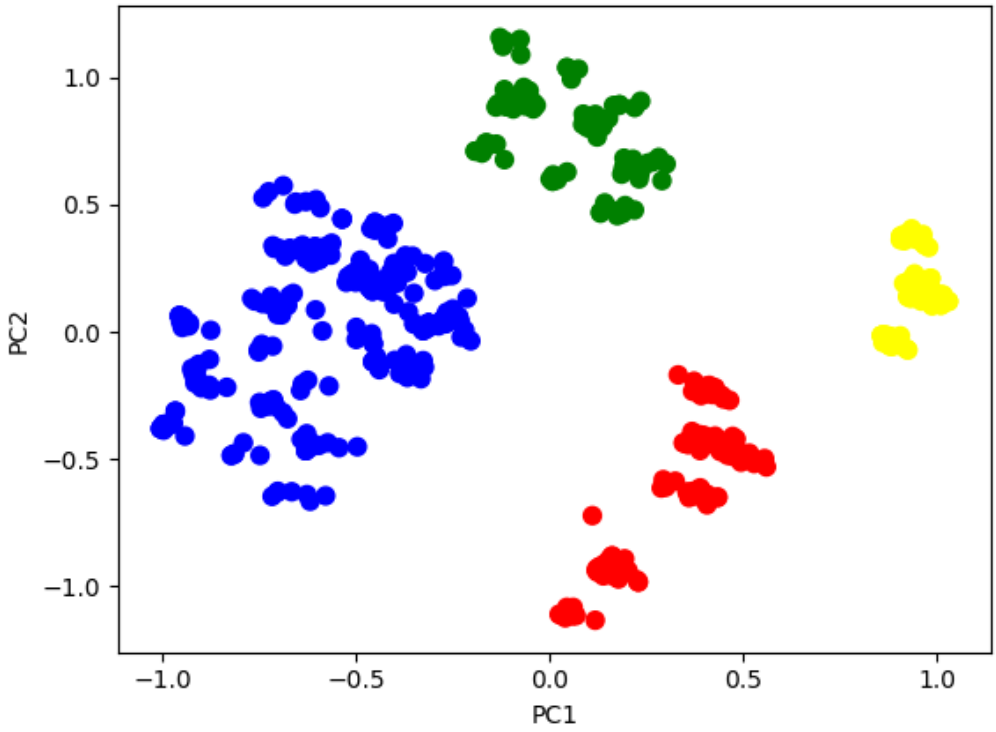

Figure 22: User data clusters

5.5.2 Visualization of characteristics

From the previous picture we cannot infer anything about the clusters characteristics. So, as we mentioned before, we need to display distributions of user variables, for every cluster, to draw conclusions about the cluster characteristics.

For examples, value distributions for each feature describing one of the clusters are shown at next figures.

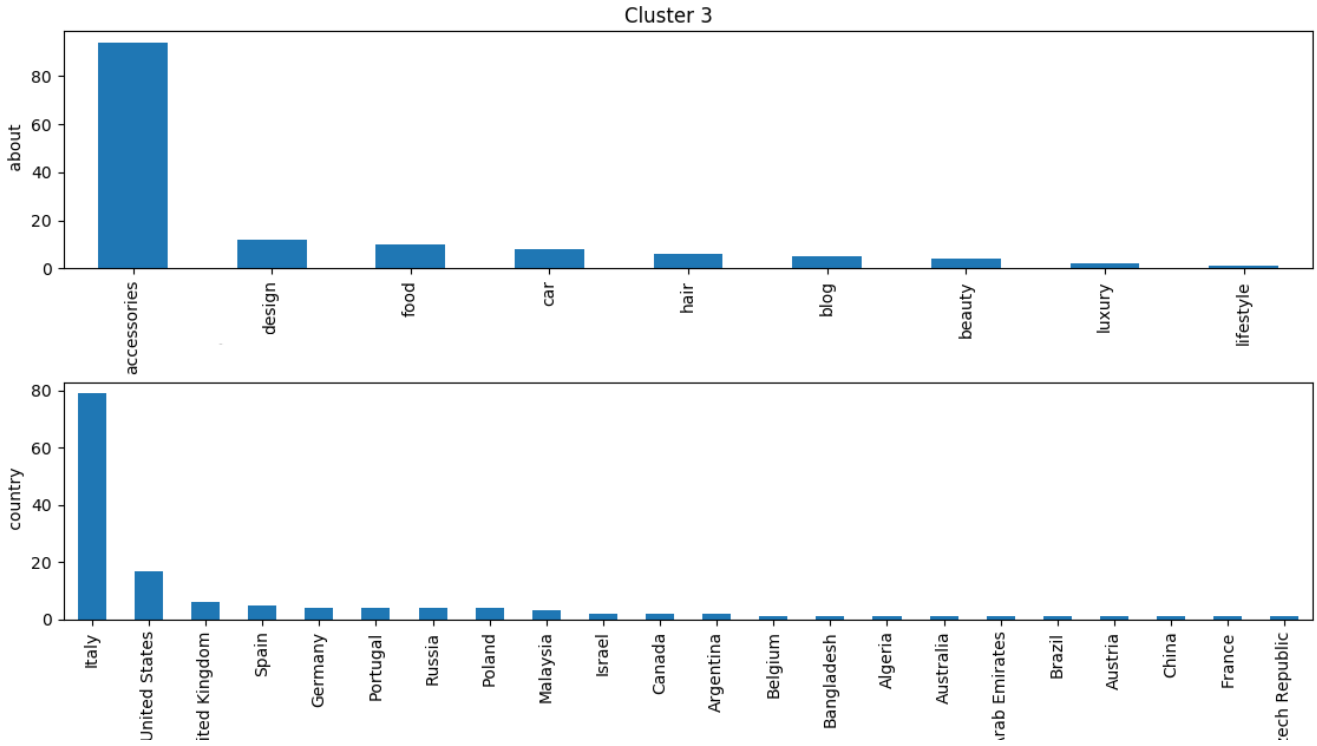

Figure 23: About and Country features distributions of user data

We can see that in cluster 3 for example users or pages are mostly coming from Italy and are dominantly related to accessories category. There are also other nationalities present but in lower number.

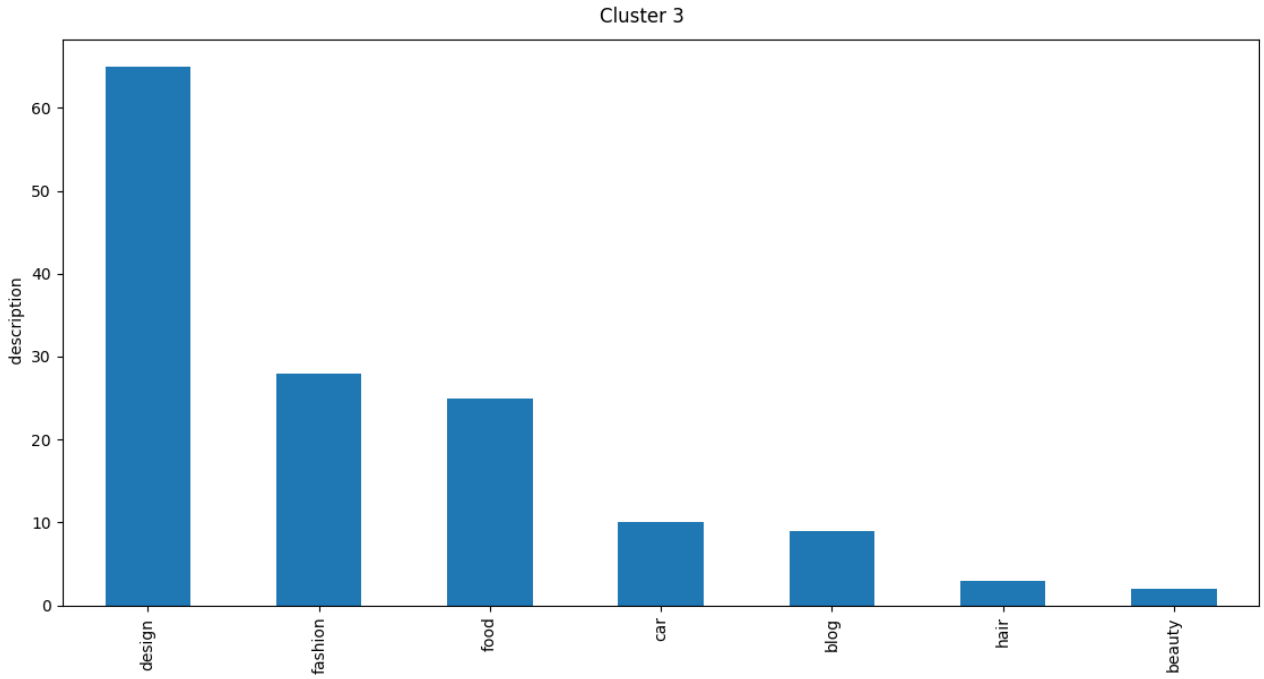

Figure 24: Description feature distribution of user data

Also, from the distribution of description feature of user data we can conclude that user/pages mostly relate to a job in the design sector.

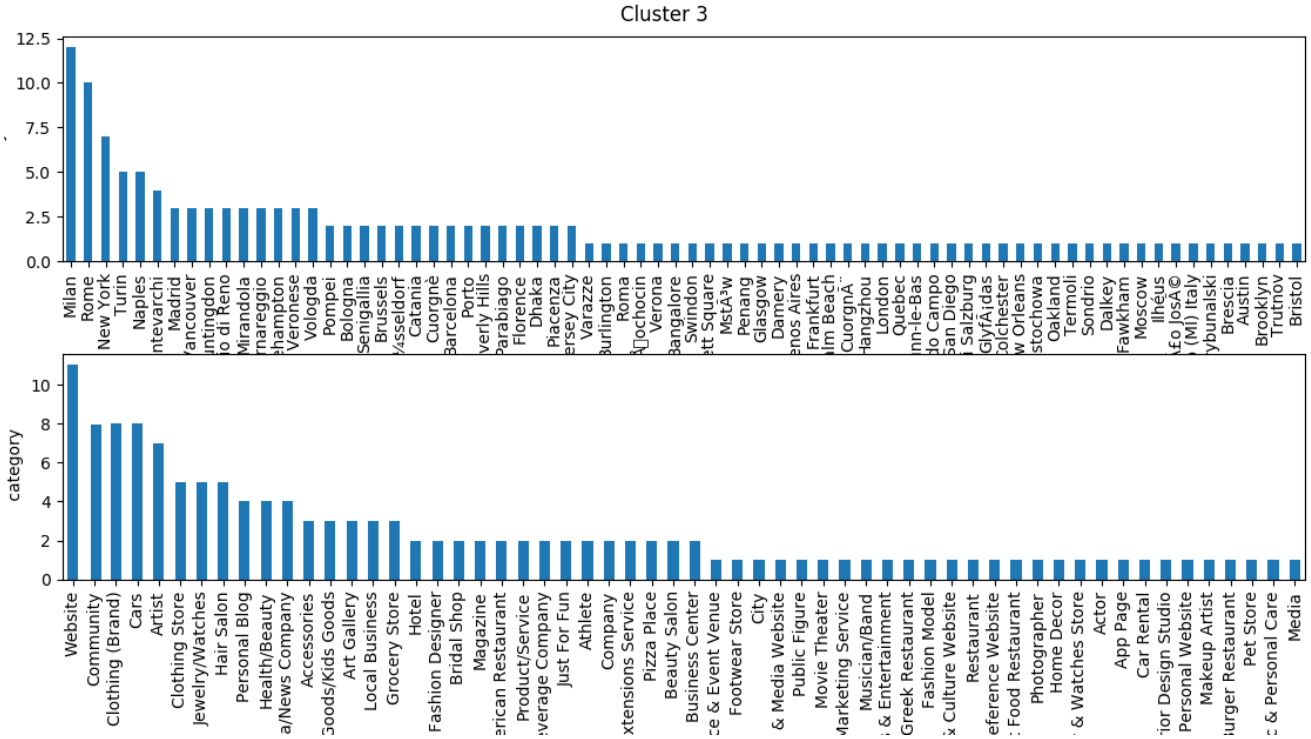

Figure 25: Category and City features distributions of user data

Finally, we look at the category and city features and observe that most users/pages from these clusters are websites coming from Milan.

While the current dataset has a low number of features, companies have more permissions on user data and their analysis thus can be more complex. The described clustering and other unsupervised techniques have an important role in the market-segmentation part of any business process. Market-segmentation is a process where the market or potential customers are grouped in finite number of groups, where each group have specific characteristics. This is done because it is neither feasible for companies to target the entire market, nor each single customer. So, companies are making marketing strategies and target-specific user groups that are the best fitting in their business plan. The main goal of marketing segmentation is to reach consumers with specific needs and preferences. It is very useful for the company in the long way because its resources are used in more effective and optimal way. Companies today are confronted with many competitors and social media analysis is one of the most important parts since it can bring advantage w.r.t competitors.

One of the advantages of our API that is not supported by currently available solutions is sentiment analysis on the clustering results. Each cluster has its own characteristics, so as users belonging to it. Having that in mind, we can determine summarized sentiment of users separately to each cluster and then compare the obtained results.

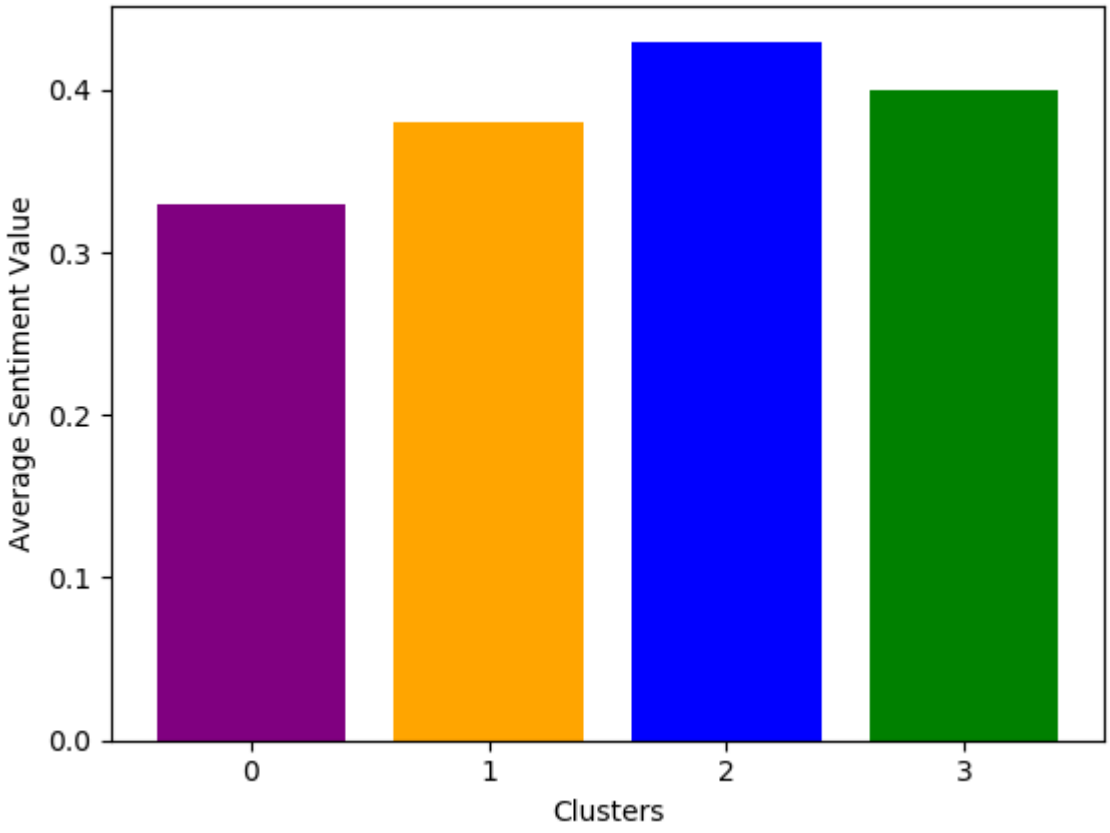

Figure 26: Sentiment analysis of clusters

In Figure 26 we can see summarised sentiment values of users for each cluster separately. We see in this example that all groups have positive average sentiment. In our dataset we see that users of cluster 3 have most positive reactions among other users. That fact represents one of the most important points in marketing segmentation process and can be used in making further conclusions in analysis.

Chapter 6

API design and development

In this chapter we present the way the API combining the analyses presented in the previous chapters is structured, how it is used and how it was developed. First, we will describe the main modules of the API and the way of use. Then we will use a top down-approach to describe the structure of the API, which means that we are going from global components description and their interaction and then we will describe each component in more detail.

6.1 Design

In today's technological world, it is very important to follow the best practices when designing your solutions. Best practices involve solutions created in optimal way, which are easy to use at the same time. So, it is available also for users who are not programmers or who are not generally technically experienced. Good practices are usually solved with nice graphical solutions. API development is different. API stands for Application Programming Interface and it is designed for programmers. API's users are other applications, it is designed to retrieve and manipulate data. APIs are used as a service they usually have single point of entry, which allow developers to manipulate remote resources in a consistent and automated way.

That is the reason why we choose the API approach to structure our project. We created a RESTful API that offers analysis as a service. It is built on top of the database systems. Its components also use external APIs to complete some inner tasks. A high-level picture of the system is presented in Figure 27. The API consists of three components: Sentiment API, Spam Filter and Clustering API. Each of those components is created in a separate package and can be used independently. However, sentiment API is using spam filter in one of its inner tasks.

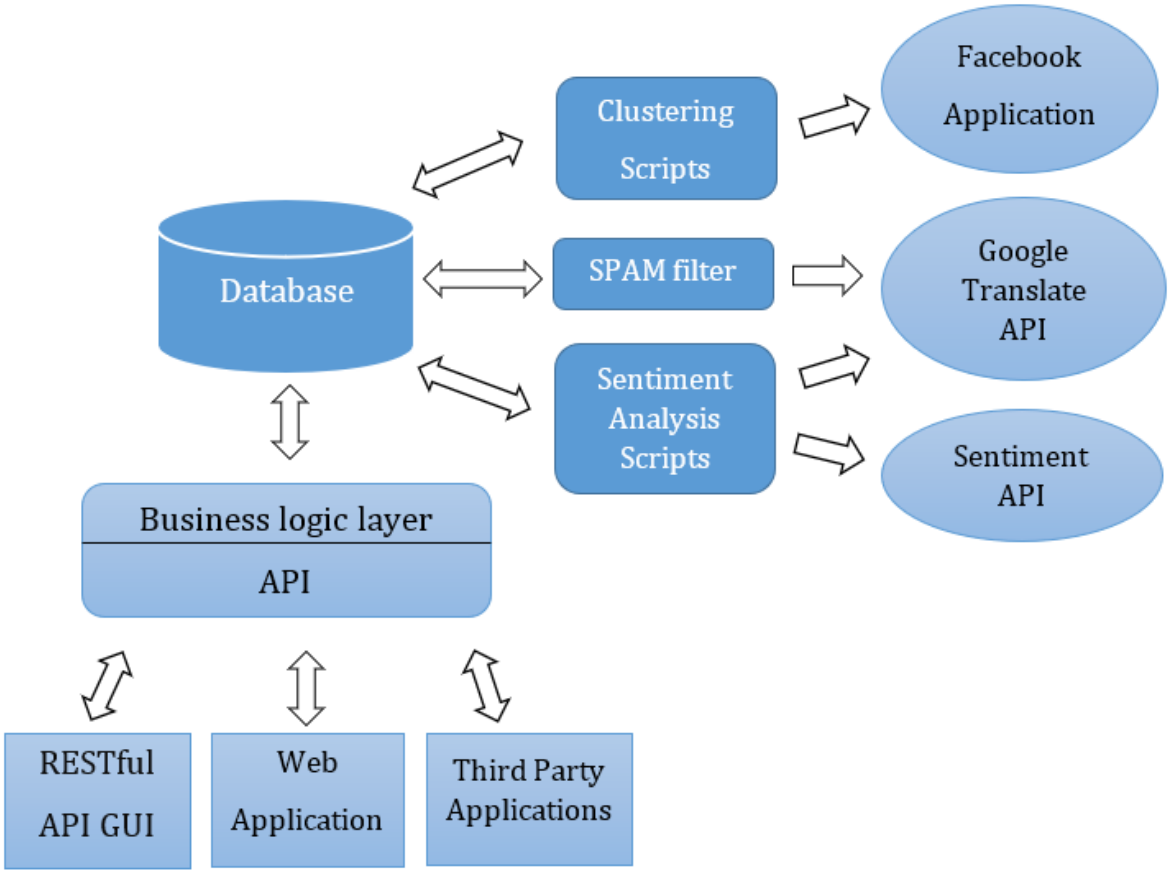

Figure 27: Social data analysis overview

6.2 Development

In this section we describe the technologies used for the development of the system. We sill specify how and where are each one of them is used. Technologies that were utilized are:

- DBMS: MySQL 5.7.14
- Scripts: python 3.6.1
- Package manager: conda 4.3.17
- APIs: Facebook API, Google Translate API
- Version control system: Git

We were provided a dump database file, where we had over 500000 comments posted by users as a reaction on posts provided by company. It is not a very large dataset, so it is mainly used as proof of concept.

In the following parts of this chapter, we describe the most important considerations and decisions we took in development phase.

6.2.1 Spam filter

Social networks or emails or blogs are places where people can interact. This is potential target for entrepreneurs and malicious users who want to make promotion in atypical way. Spam is irrelevant or unsolicited messages sent over the Internet, typically to a large number of users, for the purposes of advertising, phishing, spreading malware, etc. [Oxford dictionary definition] In the beginning, we created spam filter to be a simple function that uses regex expression for detecting links inside the text. After that, we identified that we needed a more intelligent spam filter based on supervised machine learning techniques. As described in Chapter 3, this new spam filter can detect spam based on text content.

Training and creation of new spam filter is done inside *spam_detection.py* script. First step in this process is to filter strings with links inside. Those are detected as spams using regex expression to search for links inside. After that rest of the text is tokenized and lemmatized. After dataset is pre-processed we performed holdout split into training and test set with sizes 75% and 25% respected. Next, Gaussian Naïve Bayes model is selected since it is shown very good for this purpose. The model is initialized and trained on the training dataset, before we made classification of the spams. This model as its evaluation is supported with scikit-learn library. The model is evaluated as 90% accurate.

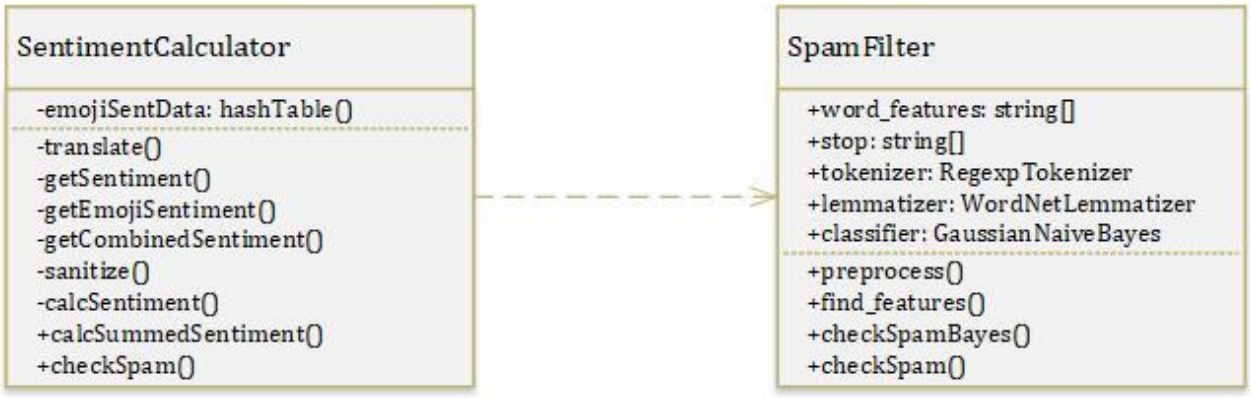

Figure 28: Sentiment calculator class diagram

6.2.2 Sentiment API

The Sentiment API module uses several components which are displayed in Figure 29. Sentiment API starts by obtaining the comments that satisfy some condition. For example, all comments left by users on specific post, or all comments left by user or group of users. After that, we are filtering actual comments from spams. For that purpose, we are using Spam filter that was previously trained.

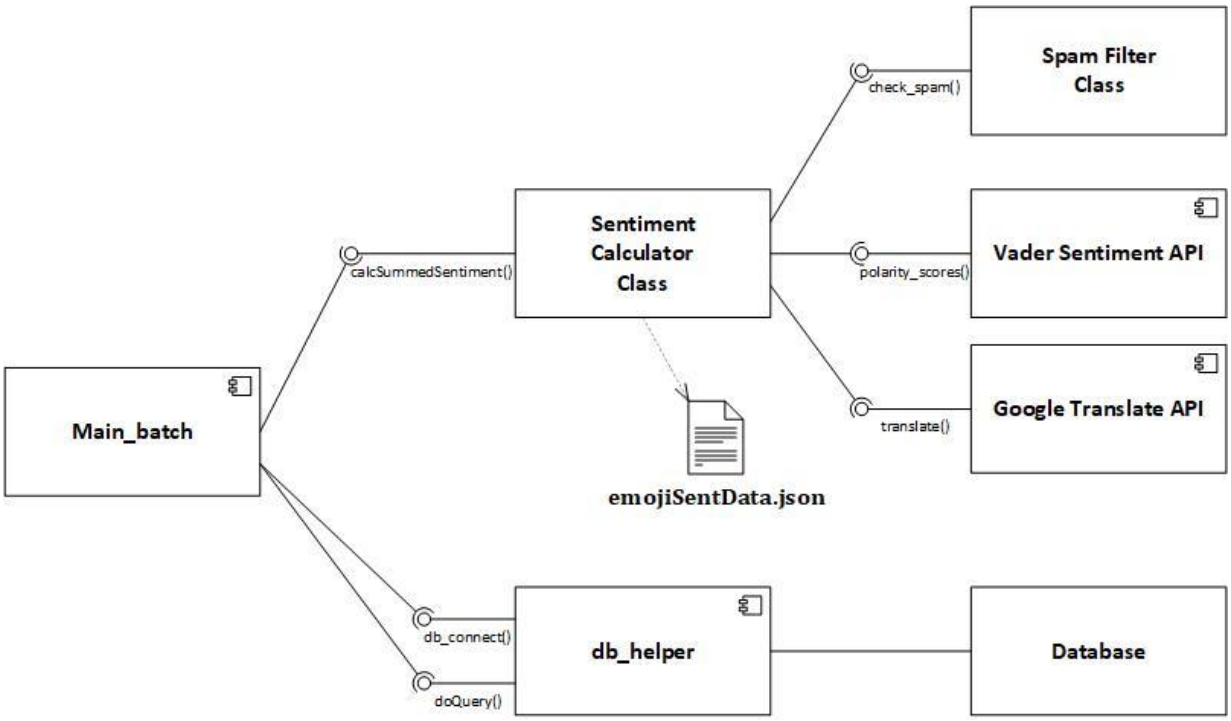

Figure 29: Sentiment analysis package components

Next, for each comment we are splitting emojis from real text. We are determining sentiment of emojis by summing up their sentiment values obtained from hash table of emojis with sentiments. Hash table with emoji sentiments is shown in Figure 30.

After that, the sentiment of the text part for every comment is processed. For that purpose, we used Google Translate API methods to translate text from various languages into English. Next, the translated text is inspected with Vader Sentiment API methods to determine the sentiment. Finally, two sentiments are combined to determine the total sentiment. It is described in more detail in Chapter 4.2.3.

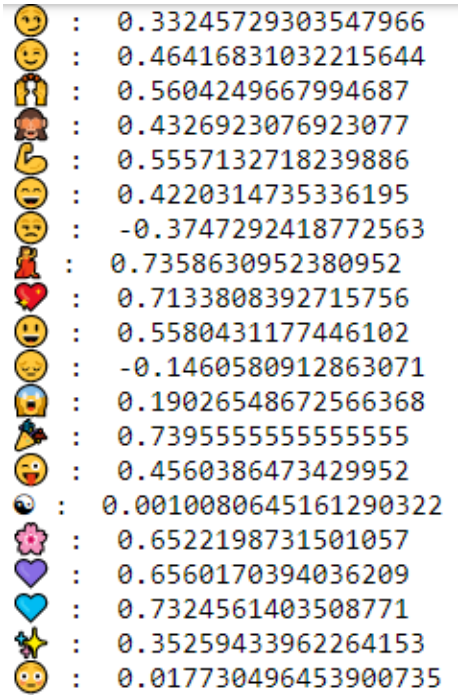

Figure 30: Emoji sentiment hash table

6.3.2 Clustering API

The Clustering API consists of many interconnected components. Two main modules are: pre-processing and clustering modules. One module is also devoted to inspecting the results of clustering where the main focus is on cluster visualizations.

The pre-processing module represents a separated package that is shown in Figure 31. Two main components, that represent an interface of the package, are *fetch user data* and *create user dataset* components. *Fetch user data* component find the id of users that we want to analyse and get the data that describes them from one of targeted social channels. Data is stored in *user_social* table of *sentiment_db* created for that purpose. In that process that component uses helper classes stored into *utils* package. *Database class* is a class that provides connection to the database. It uses singleton design pattern, to ensure unique database connection. We use this class to get relevant user ids, which afterwards we use as input to FacebookAPI class. This class is used to connect Facebook network and return data from users with provided ids.

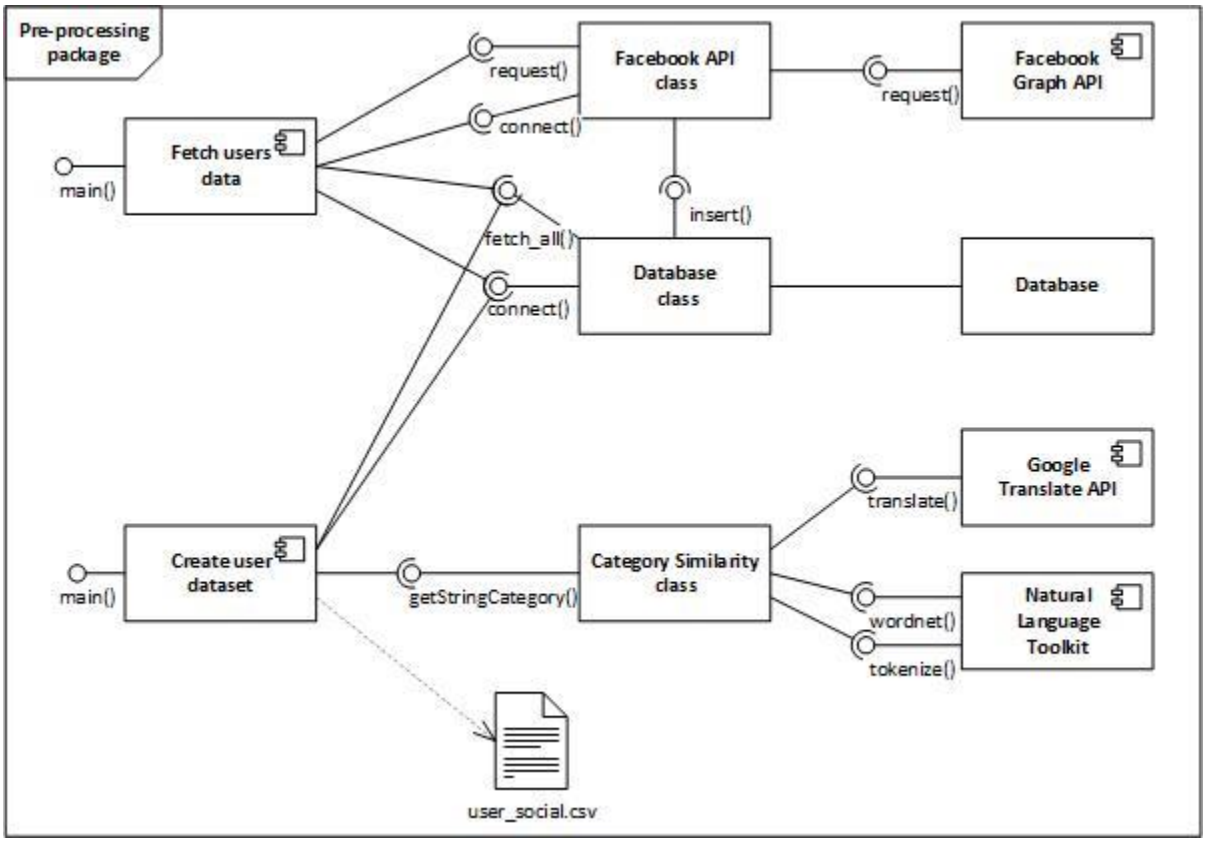

Figure 31: Pre-processing package component diagram

The *Create user dataset* component is used after user data are fetched and stored in the database. It loads data about users and modifies it so that after the process data is ready to be analysed. It uses the *Database* component to read fetched data. Database component realizes singleton design pattern to ensure one open connection for current process to database. Then the usual prep-processing steps are applied, missing data are filled in with respect to their distribution. Categorical variables are handled with integer encoding and one-hot encoding technique, where integer encoding is just enumerating categories, and replacing them to an integer number. One-hot encoding technique is performed after integer encoding is completed and enlarges dataset by adding number of columns equal to number of categories. Values of those columns are ones and zeroes, which indicates weather current instance belongs to a category or not. After columns are added, one random column is removed to prevent redundancy in data. String variables are handled using helper *Category Similarity* class. *Category similarity* class uses Google Translate API and Natural Language Toolkit component, first to perform translating string into English and then to categorise it into one of the target classes. Classes are better described in Figure 32.

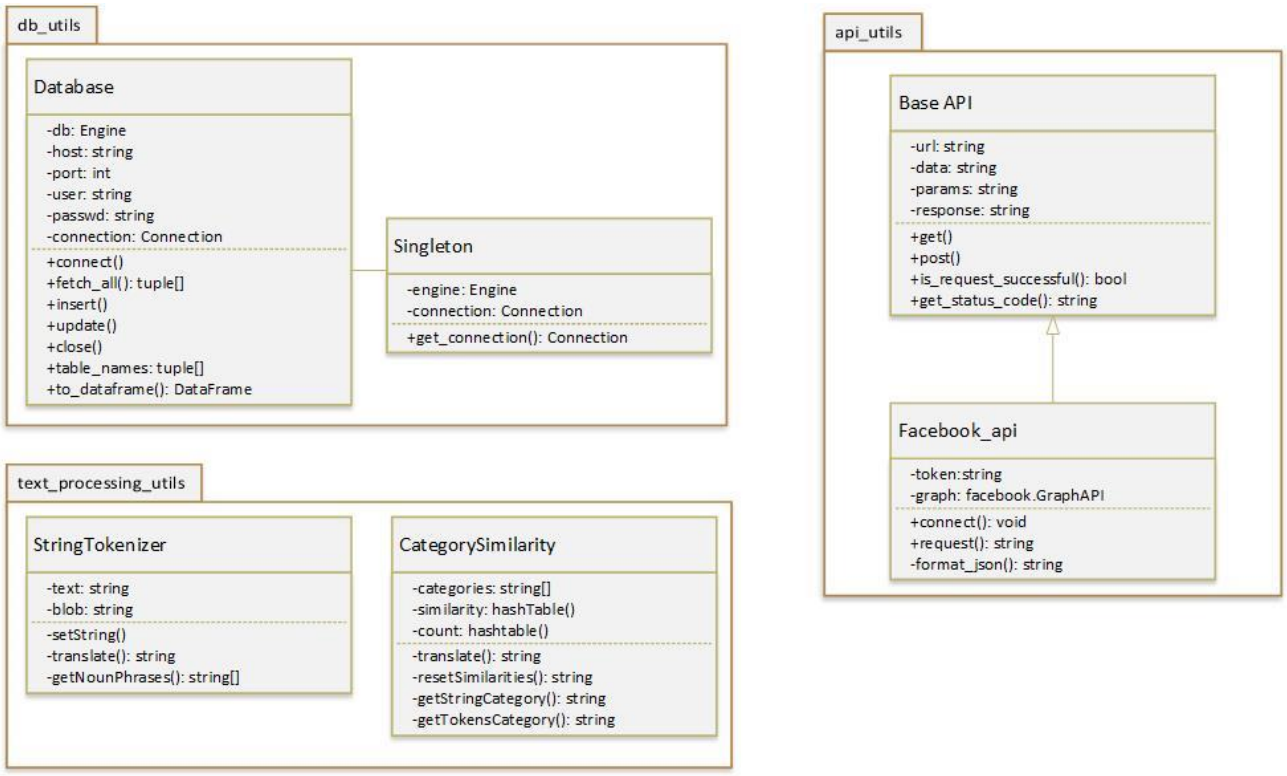

Figure 32: Pre-processing utils package class diagram

In categorization step of string variables, categories are not assumed in methods of Category Similarity class, they are previously determined. This is done with nlp_*about* component. The components included are presented in Figure 33. First, all string data are tokenized and, using a dictionary, word occurrences are counted. Top ten of most frequent relevant tokens are selected as default categories.

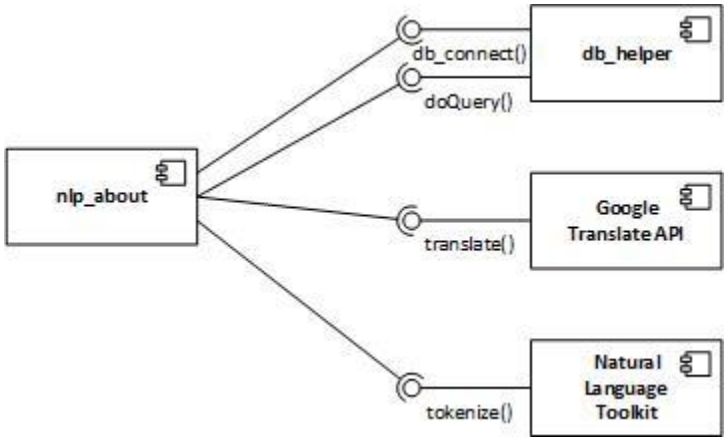

Figure 33: Component used to determine default categories

Next, the clustering step is done after pre-processing is finished. First, we load created pre-processed data and determined which are relevant features of users for the analysis. Sequence diagram of actions during the clustering process is described in Figure 34.

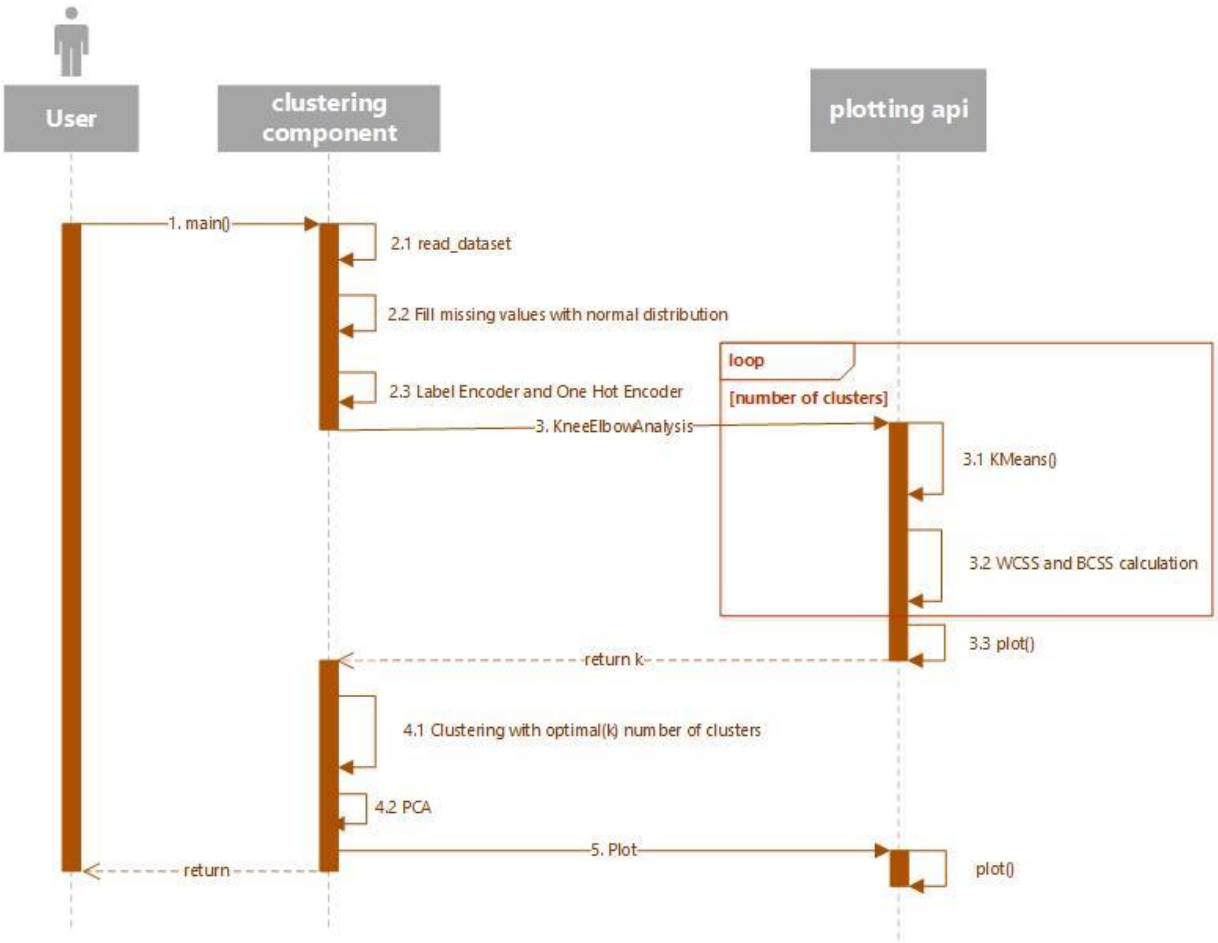

Figure 34: Clustering process

Chapter 7

Conclusion

In this final chapter, we will sum up the main reasons for developing this project. We will also describe in what aspects and how much our project contributes to them. We will also describe how our project improved existing solutions. After that, we will give a brief overview of expectations and obtained results. Finally, we will describe possible future improvements.

Current companies today are creating business plans where it is very important to be innovative and to have good prediction of future trends. This gives an advantage over the competitors. So, companies are trying to obtain as much information about their potential market share. Now, business is moved to the Internet and the main focus is on social networks and user generated data. All users are observed as customers and their behaviour is treated with great consideration. So, companies are now searching for tools that "predicts future" using the mentioned data.

Our project gives support for these kinds of problems. To be able to correctly analyse user data, we need to handle them properly. One of these problems is related to spam filtering. Spam relates to useless data that need to be removed before performing any kind of analysis. We dedicated one part of our project specifically to this topic. We created an intelligent spam filter that automatically detects and removes spams from dataset of interest. Spams with links are easily detected with simple regex expressions, but other cases need to be handled in more complex ways and that is why we decided to use supervised machine learning algorithms to solve this problem.

We have already mentioned that user reactions are very important, so companies need to find a way to use them as an advantage of social networks. Now we see that companies are very active on Facebook, Instagram, Twitter. They are posting information on new products and try to see how potential customers react. We recognized that companies need a tool to measure sentiment of users about specific product, so we built a sentiment analysis API that takes user's comments on some entities, process them and returns summarized sentiment on the commented entties. Users can be of different nationalities and thus they will most probably use their native language when leaving a comment. Moreover, now there is the possibility of using emojis or hashtags to express reactions. In other words, we have reach multi-lingua textual data that need to be managed. In the development of the Sentiment Analysis API we used an external component that detects language and translate it in English, thus ensuring consistency in the treated data. Because it is rich text, we exploit emojis to determine sentiment more precisily. So this component split data in two parts: emojis and raw text. Sentiment is calculated separately for each part and then it is summarized in optimal way.

Another point that this report indicates is the need for clustering of customers. Since, it is impossible for most of companies to target the whole market space, they adopt marketing segmentation, which is one of most important elements of creating business plans. Through clustering, users are grouped based on their characteristics, and they are identified in a group with most similarity among others. So, when a company decides for example to launch a product, it can see which groups best match particular products. In that way, companies can manage their resources in a more effective and more optimal way. In our project we used k-means clustering algorithm with initial number of clusters that was previously determined with elbow analysis technique. In existing solutions there is no support in user clustering methods and that is one problem that our project solved.

A lot can be done to improve the developed modules and make them more robust. The machine learning spam filter can be trained on a different dataset that the one we used. It would be ideal to gather data from the social network the spam filter is going to be used for, and then train it over this specific content. Also, the dataset that we used was quite small, and improved accuracy can be achieved by training the classifier against larger datasets.

Sentiment analysis can be enriched with a social network specific content that was not taken into account. For example: Facebook reactions could provide very useful information for the sentiment, or likes or Twitter hashtags.

Bibliography

[1] Hutto, C.J. & Gilbert, Eric. (2015). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014.

[2] P. Kralj Novak, J. Smailovi´c, B. Sluban, and I. Mozetic, "Emoji Sentiment Ranking," *PLoS ONE*, vol. 10, no. 12, p. e0144296, 2015.

[3] Tianran Hu, Han Guo, Hao Sun, Thuy-vy Thi Nguyen, Jiebo Luo. (2017), "Spice up Your Chat: The Intentions and Sentiment Effects of Using Emojis"

[4] W.A. Awad and S.M. ELseuofi (2011), "Machine Learning Methods for Spam E-mail Classification", International Journal of Computer Science & Information Technology (IJCSIT), Vol 3, No 1, Feb 2011

[5] Christopher M. Bishop (2006), "Pattern Recognition And Machine Learning"

[6] Paul Attewell, David B. Monaghan, Darren Kwong (2015), "Data Mining for the Social Sciences"

[7] "Importance of Social Media analytics" Available at: https://sysomos.com/2016/06/14/the-importance-of-social-media-analytics/

[8] "Already available solutions for social media analytics" Available at: https://sproutsocial.com/insights/social-media-analytics-tools/

[9] "One-hot encoding technique for categorical variables explained" Available at: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

[10] Ming-Hsuan Wu, "Opinion Mining and Sentiment Analysis", Graduate project report, Faculty of The School of Engineering & Computing Sciences Texas A&M University-Corpus Christi Corpus Christi, TX, 2014

[11] B. Pang, L. Lee (2008), "Opinion Mining and Sentiment Analysis", Foundations and Trends in Information Retrieval Vol. 2, Nos. 1–2 (2008) 1–135 2008 B. Pang and L. Lee DOI: 10.1561/1500000001

[12] J. Read (2005), "Using Emoticons to reduce Dependency in Machine Learning Techniques for Sentiment Classification"

[13] A Go, R Bhayani, L Huang (2009), “Twitter sentiment classification using distant supervision”, CS224N Project Report, Stanford, 2009

List of figures

List of tables