**POLITECNICO DI MILANO**

Department of Electronics, Informatics and Bioengineering

M.Sc. programme in Computer Science and Engineering



**Automated Analysis of Social Data**

**using Machine Learning Techniques**

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 person. So naturally, companies recognize the opportunity and try to make use of that by changing their business plans and focus to potential customers on social networks. Business is realized by company’s presence on web and producing a content that will take customer’s attention. In return users share their opinion about particular products by leaving comments on them and reacting on company’s posts.

Taking that into consideration, it is useful to have an automated way to check user reactions on products that company is offering. Also knowing types of people fallowing and leaving opinion on products can be turned into advantage for creating future business plans. For example, to predict which products can be attractive for specific user groups or to determine 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 achieve that goal.

In this thesis we have built automated for calculating sentiment of users who commented on specific company’s post along with intelligent spam filter. Sentiment analysis was done separately on text and on the emojis. For the evaluating text sentiment, we used open source API and for emojis we used table of evaluation for each emoji. Spam filter was designed using supervised machine learning techniques to determine spam, not just by searching URL patterns in comments, but also to determine the spam by checking text content. We have also built clustering module which uses unsupervised learning techniques on user data and visualizing characteristics for each discovered group. Finally, in last chapter is defined how previous models can be used together in an API to evaluate success of company’s posts.

Contents

[1 Introduction 9](#_Toc499602041)

[1.1 Structure 10](#_Toc499602042)

[2 State of Art 11](#_Toc499602043)

[2.1 Need for analysis of social data 11](#_Toc499602044)

[2.2 Applications of social data analytics in various companies 11](#_Toc499602045)

[2.3 Methods and tools used for social data analytics 12](#_Toc499602046)

[3 Automated sentiment analysis of social network content 13](#_Toc499602047)

[3.1 Social network content 13](#_Toc499602048)

[3.1.1 Database 14](#_Toc499602049)

[3.2 SentimentCalculator 14](#_Toc499602050)

[3.1.2 Text Sentiment 15](#_Toc499602051)

[3.1.3 Emoji Sentiment 16](#_Toc499602052)

[3.1.4 Combined Sentiment 18](#_Toc499602053)

[3.3 Usage and results 19](#_Toc499602054)

[4 Machine Learning Spam Filter 21](#_Toc499602055)

[4.1 Comparison of machine learning algorithms for spam filtering 21](#_Toc499602056)

[4.2 Naïve Bayes classifier method 22](#_Toc499602057)

[4.3 Naïve Bayes implementation of spam filter for social data 25](#_Toc499602058)

[4.3.1 Dataset 25](#_Toc499602059)

[4.3.2 Bag of Words model 25](#_Toc499602060)

[4.3.3 Tokenization 26](#_Toc499602061)

[4.3.4 Stop Words 27](#_Toc499602062)

[4.3.5 Stemming and Lemmatization 28](#_Toc499602063)

[4.3.6 Training the Naïve Bayes classifier 29](#_Toc499602064)

[5 Unsupervised approach to Social Data Analysis 32](#_Toc499602065)

[5.1 Fetching data from social networks 32](#_Toc499602066)

[5.2 Pre-processing 33](#_Toc499602067)

[5.2.1 Exploratory data analysis 33](#_Toc499602068)

[5.2.2 Missing data 34](#_Toc499602069)

[5.2.3 String data 35](#_Toc499602070)

[5.2.4 Categorical data 35](#_Toc499602071)

[5.3 Unsupervised learning techniques 35](#_Toc499602072)

[5.4 Elbow analysis 37](#_Toc499602073)

[5.5 Visualization of clusters 38](#_Toc499602074)

[5.5.1 Principal Component Analysis 38](#_Toc499602075)

[5.5.2 Visualization of characteristics 40](#_Toc499602076)

[6 API design and development 43](#_Toc499602077)

[6.1 Design 43](#_Toc499602078)

[6.2 Development 44](#_Toc499602079)

[6.2.1 Spam filter 44](#_Toc499602080)

[6.2.2 Sentiment API 44](#_Toc499602081)

[6.3.2 Clustering API 45](#_Toc499602082)

[7 Conclusion 49](#_Toc499602083)

[Bibliography 50](#_Toc499602084)

[List of figures 51](#_Toc499602085)

[List of tables 52](#_Toc499602086)

**Chapter 1**

# Introduction

Analysis of social network content is difficult because conversation on social networks differs in many ways from normal conversation. Contents are rich with emojis, hashtags, mentions and spams which the one need to filter and process along with raw text to find the information behind it.

By analysing social networks data about certain products, brands or certain campaign 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 opportunity to user to analyse contents using sentiment analysis to determine sentiment of users on specific product, supervised machine learning algorithms within spam filter module to efficiently detect and remove spams from dataset and finally clustering module that discover user groups and their characteristics which can be used in making future predictions.

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, product, etc. is positive, negative, or neutral [Oxford dictionary definition]. It is combined with emoji sentiment evaluation in a way that sentiment analysis is done on the raw text using open source API while the emoji’s sentiment is evaluated using sentiment tables. Final sentiment is defined as specific combination of those two sentiments.

Spam filter module is equipped with two components. First part is done as text processor using regex expressions to detect links inside the text. Second part is trained Naïve Bayes machine learning model for detecting spams by checking the text context and represents more intelligent way of doing it.

Clustering module is created as unsupervised machine learning model, that taking users finds optimal number of user groups. It is also equipped with visualizing part that displays specific characteristics for each user group.

Finally, previously described models used together can be used in one complete API which will make predictions on successfulness of company’s posts left on social networks. That API can be used as a part in any company’s business intelligence application.

## 1.1 Structure

* Chapter 2 is about the reasons for having tools for analysing social networks within company’s applications, the advantage of having it and how to use it. We will mention examples where we can use it.
* Chapter 3 is dedicated for automated sentiment analysis of social network content. Here we are going to talk about sentiment calculation, pre-processing of emoji’s and their sentiment, and combining text emoji sentiment. We are going to describe our approach and architecture.
* Chapter 4 is about machine learning approach to create spam filter. This includes pre-processing, training the dataset and evaluation of classifier.
* In chapter 5 we will talk about unsupervised approach to social data analysis. We will go through all steps of this process which includes: fetching the data from social networks, pre-processing, unsupervised learning techniques, elbow analysis and visualization of clusters characteristics.
* Chapter 6 is reserved for describing API which will use previously mentioned modules in order to analyse custom social data.
* Chapter 7 will finalize the purpose of this project and present the conclusion. We will also mention 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 three sections, each of them trying to emphasize the need for analysis of social data and their impact to company’s business.

1. Need for analysis of social networks data
2. Applications of social data analytics in various companies
3. Methods and tools used for social data analytics

## 2.1 Need for analysis of social data

Social media gives businesses an unprecedented opportunity for connecting with customers and prospects. While there are numerous social networks that provide you with a vast array of tools for providing customer service, explaining how your products work and much more, it’s 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 protentional 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, company can have model of a user and can predict weather new product will be successfully accepted and attractive to potential customers.

## 2.2 Applications of social data analytics in various companies

Social data analytics can be applied in many types of companies. Especially the ones which 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 successfulness 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 to see which types of user groups their followers belong to make wright predictions in the future. Fashion companies can use it to decide when is the best way to lunch specific products. It also depends are the followers young people like students, mid-age or older people, which of two genders are reacting better and what to do to stimulate their better reaction.

## 2.3 Methods and tools used for social data analytics

Some of the most famous tools that have been used for social data analytics are Sprout social, Buzzsumo and Google analytics. 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 your 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 your website performs on social media. For instance, if you want to see how many shares your latest blog post received on Facebook and Twitter, Buzzsumo can provide you with that data. Buzzsumo will not only show you the number of shares for each piece of content, but it also shows you 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,” Google Analytics is one of the best ways to track social media campaigns and even help you measure social return on investment. You likely already have GA setup on your website to monitor and analyse your traffic.

All mentioned tools use statistical techniques for analysing the social network data. While they are analysing user reactions, they don’t support analysing contents left by users, such as comments and posts.

For that part it is useful to have some machine learning techniques for analysing text context and that is what we recognize and try to create complete API that will include all modules together.

**Chapter 3**

# Automated sentiment analysis of social network content

## 3.1 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.

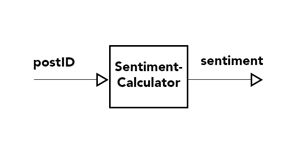


Fig 3.1. Overview of the Sentiment module

### C:\Users\Pavle\AppData\Local\Microsoft\Windows\INetCache\Content.Word\db_schema.pngDatabase

Fig 3.2. Social Media Channels Database

The contents of the social media are stored in a database that has tables like, comments, posts, social media channels, users etc. The one that we are interested is comments, which holds content of the comment (including text and emoji encoded in extended utf-8), user that posted it, corresponding postId and other meta-data. The input in to the SentimentCalculator module in this case is the postId with which we can fetch all the corresponding comments.

## 3.2 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 array of comments from a certain post and the result is 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)

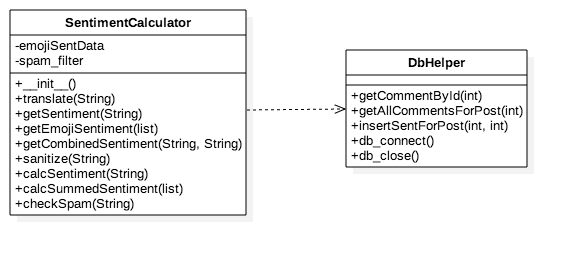


Fig 3.3. SentimentCalculator class diagram

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 importan to note that SentimentCalculator can be used as a generic module for different kind 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).

### 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 unidimensional measure of sentiment for a given sentence. Calling 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. So as a preprocessing step, each comment is translated using google translate (googletrans Python package).

### Emoji Sentiment

Emoji sentiment is calculated using emoji sentiment data stored in json format based on the work of based on the work 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

},

...

]

Fig 3.4. Emoji sentiment datum

Properties of an emoji sentiment datum explained:

* sequence (original)

normalized code point sequence (sequence without any variation selector or modifier applied) e.g. 1F602; use it for mapping the sentiment datum to a specific (emoji) unicode character or connecting it with further meta data (e.g. [unicode-emoji-data](https://www.npmjs.com/package/unicode-emoji-data), [unicode-emoji-annotations](https://www.npmjs.com/package/unicode-emoji-annotations) or [emoji-datasource](https://www.npmjs.com/package/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.

### 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:

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.

## 3.3 **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 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.

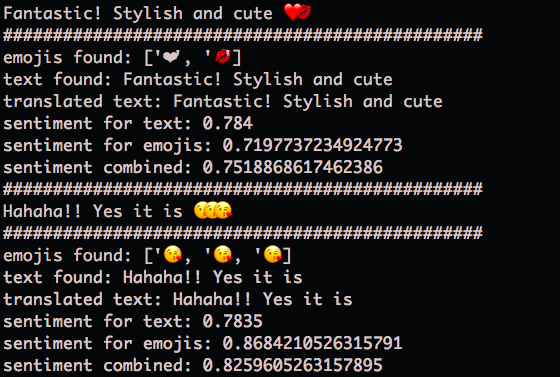


Fig. 3.5. SentimentCalculator output

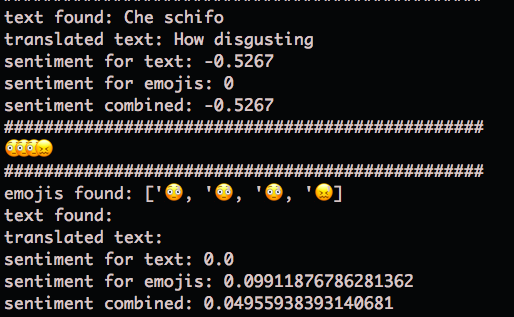


Fig. 3.6. SentimentCalculator output example for negative comments

There is also a batched sentiment calculation, that 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 4**

# 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 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 paragraph we are going to justify why we decided to use the Naïve Bayes approach for detecting whether some social content is spam or not. But first let’s review briefly some of the machine learning algorithms that are often used for this purpose.

## 4.1 **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**).

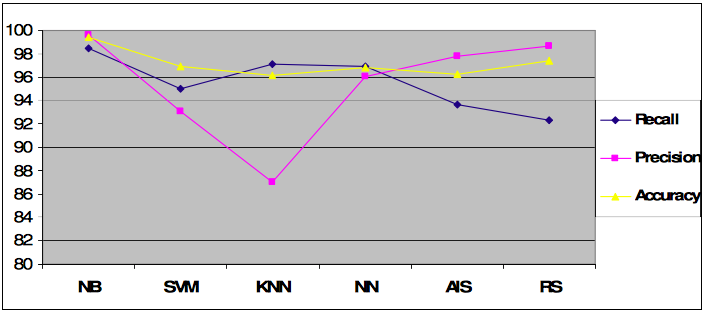


Fig 4.1 Comparison of machine learning algorithms for spam classification

In term 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, while in term 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 percent, and finally we can find that the recall is the less percentage among the six classifiers while the **Naïve Bayes** still has the highest performance but considered low when compared to precision and accuracy while the **rough sets** has the worst 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.

## 4.2 Naïve Bayes classifier method

Naive Bayes classifiers 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.

The posteriori 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 posteriori probability can be written as:

Where is the feature vector of sample *i*, and is the notation for the class *j****.*** is the probability of observing feature, given that the sample belongs to the class .

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 from 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 **x**. Thus, given a d-dimensional feature vector **x**, the class conditional probability can be calculated as follows:

Here,  simply means: “How likely is it to observe this particular pattern **x** given that it belongs to class ?” 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:

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

To illustrate this concept with an example, let’s 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 the message is spam” — the probability of encountering “world” given the message is 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:

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  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 is simply:

Classify sample  as   if  else classify the sample as , 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 implementation of spam filter, we can proceed with an in-detail 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.

## 4.3 Naïve Bayes implementation of spam filter for social data

### 4.3.1 Dataset

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

Collection consists of 1956 rows of comments on various YouTube video. From those there are exactly 1005 comments that have been 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.

### 4.3.2 Bag of Words model

One of the most 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:

* D1: “Hey, http://believemefilm.com”
* D2: “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**](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.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **hey** | **katy** | **perry** | **reminds** | **me** | **of** | **a** | **tiger** | **http://believemefilm.com** |
|  | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
|  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

**Fig 4.2.**Bag of words representation of two sample documentsD1andD2.

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 is 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 that was used is called NLTK library for Python3.

### 4.3.3 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.

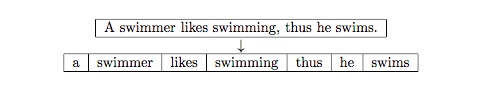
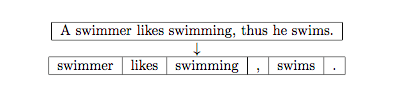


Fig 4.3. Sentence Tokenization

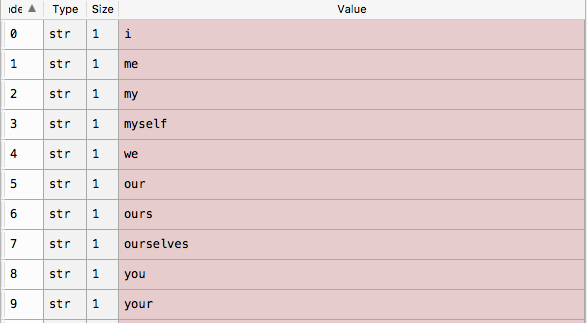
### 4.3.4 Stop Words

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.



4.4. Stop words removal

Stop words that we used are the word defined in the *stopwords* corpora of NLTK library. Some of them are:



4.5. 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.

### 4.3.5 Stemming and Lemmatization

Stemming describes the process of transforming a word into its root form. The original stemming algorithm was developed my 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.

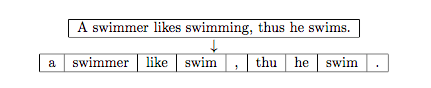


Fig. 4.6. Stemming

 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.

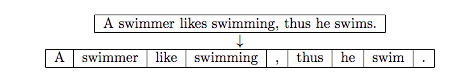


Fig. 4.7. Lemmatization

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.

### 4.3.6 Training the Naïve Bayes classifier

Not 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 of spam filter classifier. As was said earlier, since the vocabulary in this case is huge, we only chose to use first 300 most frequent words used in the dataset. Some of these words are:



Fig 4.8. Most frequent words from the dataset

Every row is transformed to 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](https://en.wikipedia.org/wiki/Supervised_learning) one (in [unsupervised learning](https://en.wikipedia.org/wiki/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).

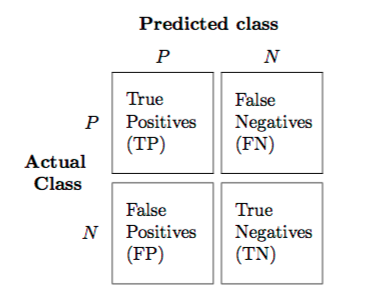


Fig. 4.9. Confusion Matrix

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

|  |  |  |
| --- | --- | --- |
|  | Positive | Negative |
| Positive | 203 | 0 |
| Negative | 30 | 93 |

There are several scores that are relevant for evaluation of the classifier that 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 approx. 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 5**

# Unsupervised approach to Social Data Analysis

Social data represents data about the users like his gender, city, description, date of birth etc. Those data can be used to detect similarities between the users. Some users are more similar and can be identified in the same group. 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 do analysis of user data, first we need to have that data. User data are present on social networks but to gather it, we need to have right permissions. For the purpose of this project we were provided a dataset which included comments, posts, social channels, brands and desired social networks with fallowing relationships:

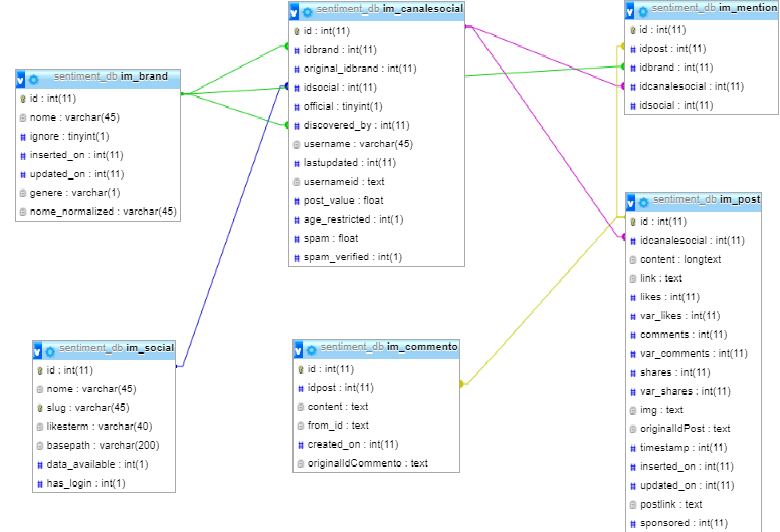


Fig1 Sentiment\_db

So, from the graph we see that for every post from *im\_post* table, we can fetch all comments that are located in table *im\_commento*. In Chapter 2 is explained how to determine summarised sentiment of all users about the post. Now, we are going further buy determining what are user groups that comments, we are interested in their characteristics and behaviour of each group. To get user data we need to use API for each social network that we are interested in. We will explain how Facebook API is used for this purpose and for the rest it is very similar procedure.

Facebook Graph API requests user authorization access before applying search. User connects to Facebook API by sending access token and to get it user need to register at *developers.facebook.com* and to register the application that will use the search. After successful authorization then user 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 it and that is explain in next part of this chapter. The result is return 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 over 10 years. Get Daily Inspiration from Me & My Team"

}

Listing x.x: FB API response

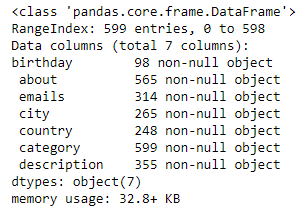
## 5.2 Pre-processing

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

In every project related to data science and machine learning 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 etc. In the next picture there is example from user data who commented on specific post:



Listing x.x: *user\_social* info

 Also, data types can be numeric, categorical or string variables. Here we must pay great attention to string variables, since they are unstructured rich 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 example on next dataset.

Figure x.x: *user\_social* 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 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 threw all rows and calculate similarity between the tokens and categories, and select most similar category. In that way we did 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 “pet” variable with the values: “dog” and “cat“.
* A “color” variable with the values: “red“, “green” and “blue“.
* A “place” variable with the values: “first”, “second” and “third“.

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, “red” is one, “green” is two and “blue” 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 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 outputted 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 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 complexity of the algorithm which is time exhausting for big datasets.

**5.3.2 K-Means**

K-means clustering is other type that we applied in our project. Before we use it, we need to specify number of clusters that we expect to have. In start, random centroids which number we specified are initialized and other data points are assigned a cluster which is the closest centroid. Algorithm is specified in fallowing figure:

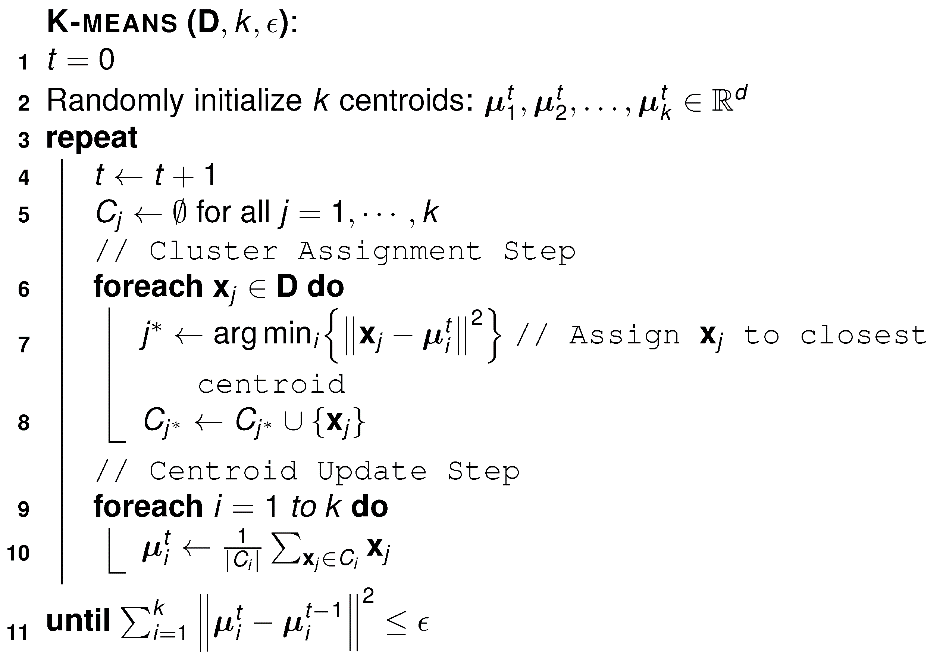


Fig x.x: K-Means agorithm

Advantage of this algorithm is that it is computationally fast algorithm and also it produces tighter clusters, especially if the clusters are globular. 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. Number of clusters was determined using Elbow analysis method which will be explained in fallowing subchapter.

## 5.4 Elbow analysis

As we mention before we have chosen K-means clustering technique to inspect our dataset. It had the best performance, but we need to find 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):

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

* Between-cluster sum of squares (BSS):

where µ is the centroid of the whole dataset

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

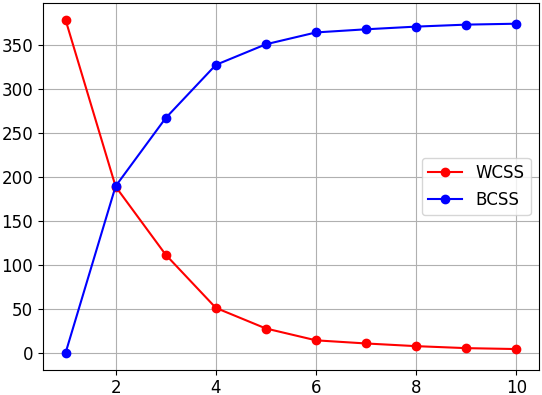


Figure x.x: Knee for K-means clustering

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 point representing four clusters, WSS is not changing much i.e. it is stabilized. So, in our case 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. We need to understand the characteristics of each cluster. In our project all the variables were categorical, so we decided to visualize distribution of values for each feature labelled by specific cluster.

### 5.5.1 Principal Component Analysis

Principal component analysis (PCA) is technique used for dimensionality reduction, data compression, feature extraction and data visualization. PCA represents orthogonal projection on a lower dimensional linear space, which is also known as principal subspace, such that the variance of the projected data is maximized.

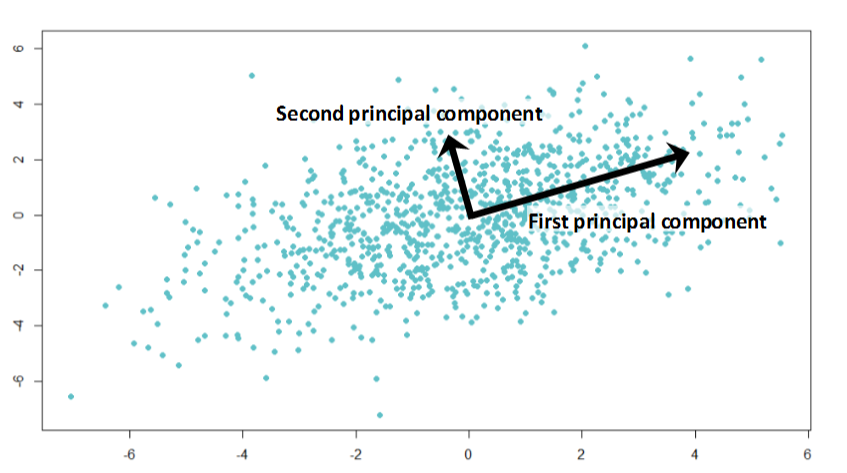


Figure x.x: Example of PCA

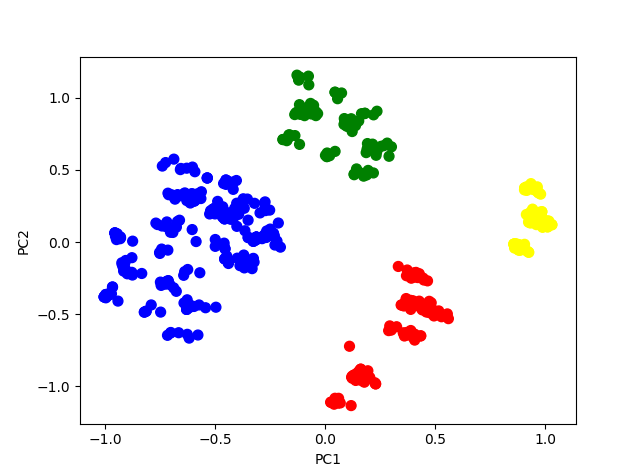
In our project we used PCA for dimensionality reduction of multidimensional user dataset to be able to visualize clustering results. Clustering of dataset is shown in next figure, where the dataset is clustered into 4 distinct groups:

Figure x.x: User data clusters

### 5.5.2 Visualization of characteristics

From 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 make conclusion about cluster characteristics.

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

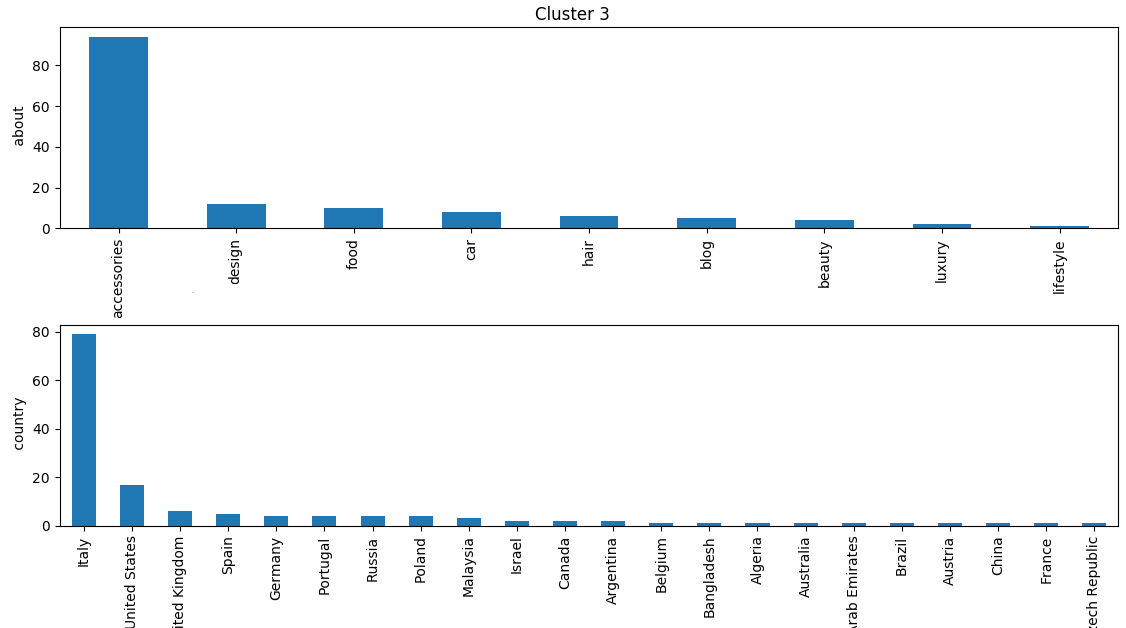


Figure x.x: About and Country features distributions of user data

We can see that cluster3 users or pages are mostly coming from Italy and are dominantly related to accessories category.

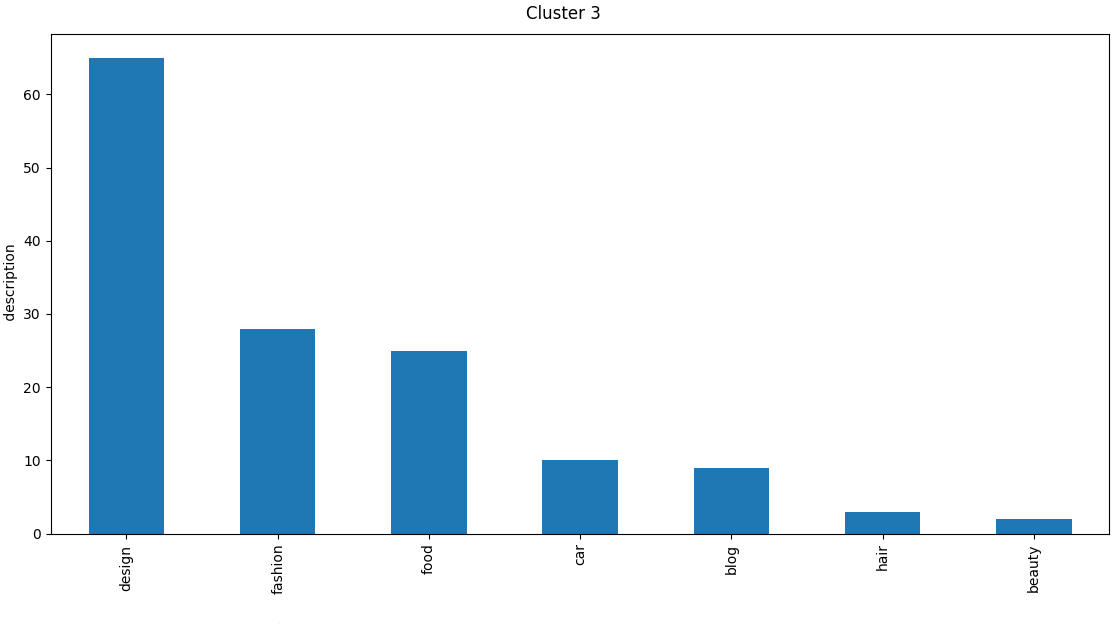
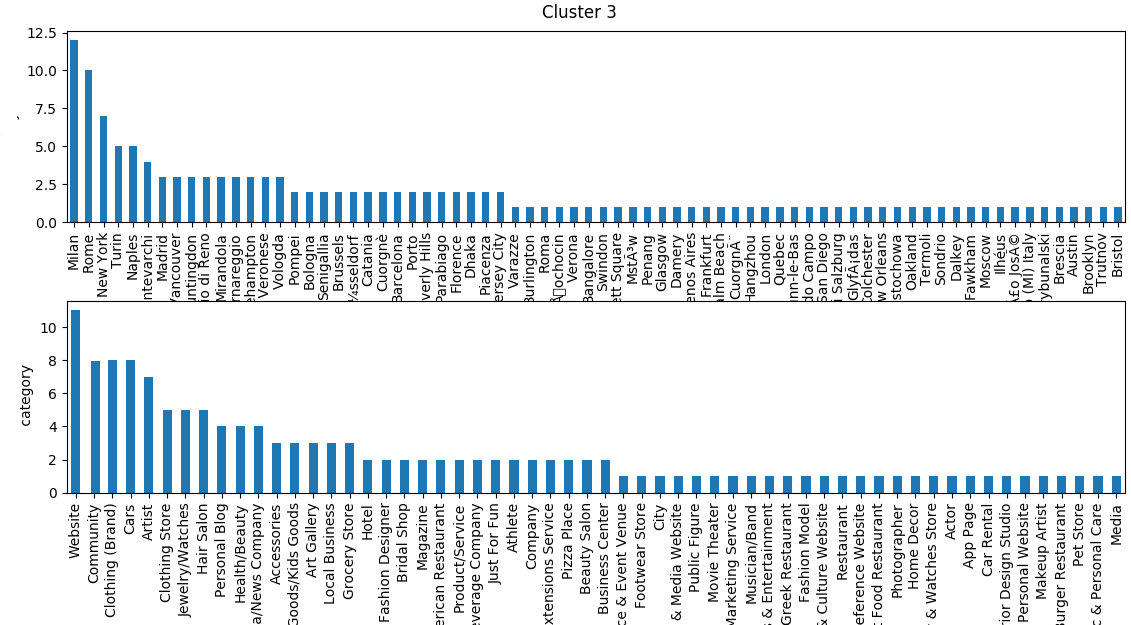


Figure x.x: Description feature distribution of user data

Also, from distribution of description feature of user data we can conclude that user/pages are mostly describe their work as design.

Figure x.x: Category and City features distributions of user data

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

**Chapter 6**

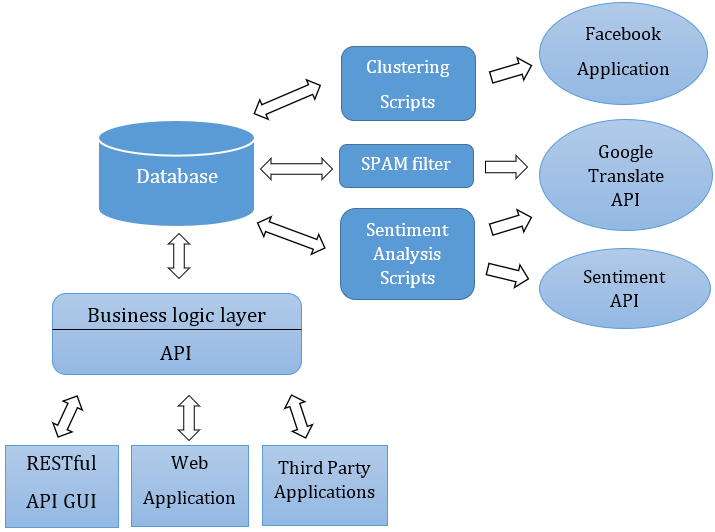
# API design and development

In this chapter we are going to present the way API is structured, how it is used and how it was developed. First, we will describe parts of API and the way of use. Then we will use top down approach to describe 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 technology world it is very important to fallow the best practices when designing your solutions. Best practices involve solutions created in optimal way which are easy to use in 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. It 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 consisted and automated way.

That is idea why we choose API approach to structure our project. We created RESTful API which offers analysis as a service. API is built on the support of the database systems. API components are also using external APIs to complete some inner tasks. High level picture of the system is presented in Figure x.x. The API for social media analysis we created is consisted of three components: Sentiment API, Spam Filter and Clustering API. Each of those components are created in separate packages and can be used independently. However, sentiment API is using spam filter in one of its inner tasks.

Figure x.x: Social data analysis overview

## 6.2 Development

In this section we are going to describe technologies used in development process 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

### 6.2.1 Spam filter

### 6.2.2 Sentiment API

### 6.3.2 Clustering API

Clustering API is consisted of many interconnected components. Two main parts are: pre-processing and clustering part. After that we will also describe one part on inspecting the results of clustering where the main focus is on cluster visualizations.

Pre-processing part represent separated package which is shown in Figure x.x. Two main components, that represent an interface of the package are *fetch user data* and *create user dataset* components. *Fetch user data* component find id of users that we want to analyse and get the data that describes them from one of targeted social networks. 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 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.

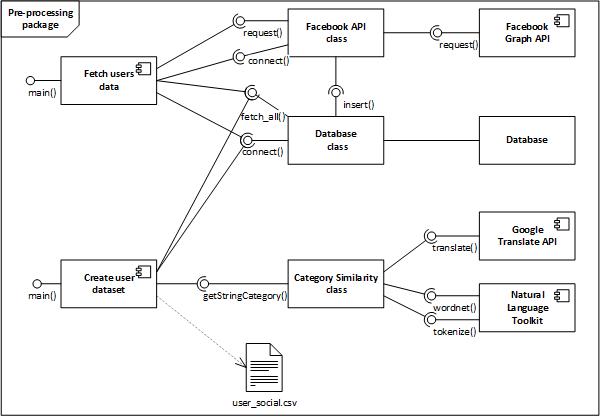


Figure x.x: Pre-processing package component diagram

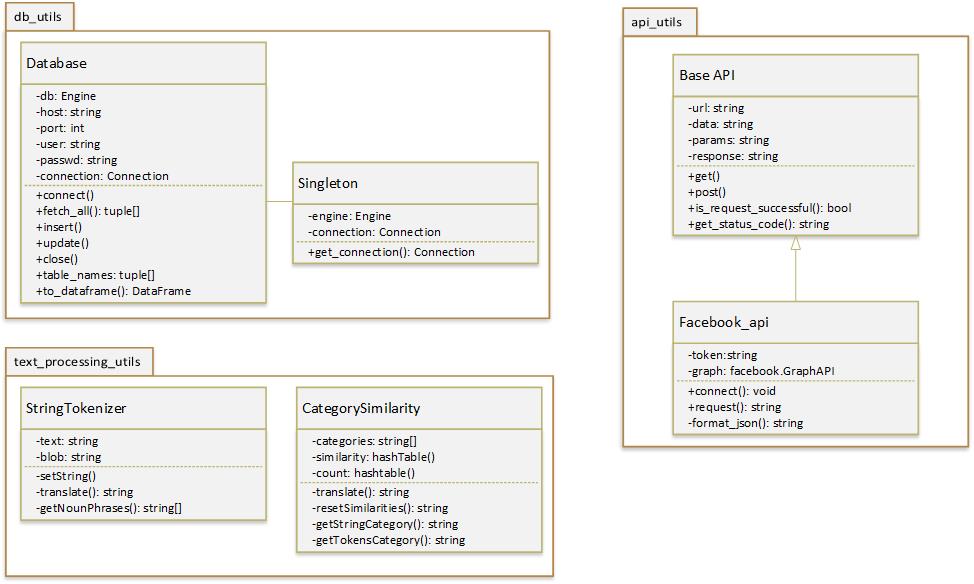
*Create user dataset* component is used after user data are fetch and stored in the database. It loads data about users and modify it in a way that after the process data is ready to be analysed. It uses *Database* component to read fetched data. Then usual prep-processing steps are applied, missing data are filled in with respect to their distribution. Categorical variables are handled with integer encoding and one-hot encoding technique. String variables are handled using helper *Category Similarity* class. *Category similarity* class uses Google Translate API and Natural Language Toolkit component to perform translating string into English and to categorise it into one of target classes. Classes are better the described with class diagram on Figure x.x.

Figure x.x 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 x.x. First all string data is tokenized and using dictionary, word occurrences are counted. Top ten of most frequent relevant tokens are selected as default categories.

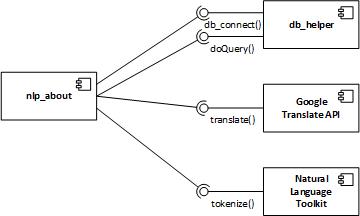


Figure x.x: Component used to determine default categories

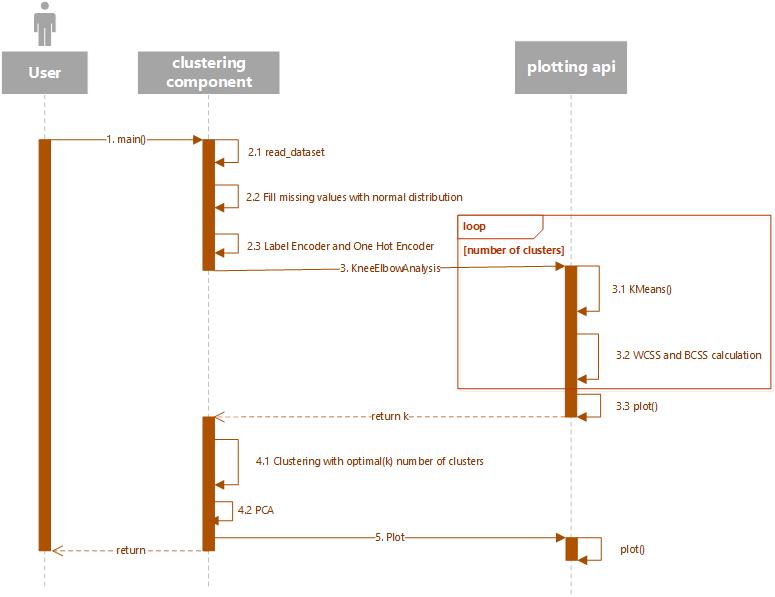
Next, clustering step is done after pre-processing is finished. First, we load created pre-processed data and determined which are relevant features of users for the analysis. Sequence diagram of actions during the clustering process is described in Figure x.x.

Figure x.x Clustering process

**Chapter 7**

# Conclusion

# 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

# List of figures

# List of tables