FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Analysis of discussion comments and their authors in social media
Student:	Martin Koucký
Supervisor:	Ing. Daniel Vašata, Ph.D.
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	Until the end of winter semester 2021/22

Instructions

The aim of the work is to analyze the discussion comments and their authors in social media using machine learning methods.

1. Try to find at least two suitable data sources that allow downloading the discussion contributions and respective comments. Focus on the ability to identify individual users.

2. Review state-of-the-art methods of the discussion comment analysis in social media. The primary aim is to analyze the users in an unsupervised way and try to identify the anomalous ones.

3. Review modern approaches to NLP and, in particular, the use of vector representation of words/texts.

4. Propose and perform the analysis of the downloaded discussion comments and users. Focus on the detection of anomalous behavior. Comment on results.

References

Will be provided by the supervisor.

Ing. Karel Klouda, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 27, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Analysis of discussion comments and their authors in social media

Martin Koucký

Department of Applied Mathematics
Supervisor: Ing. Daniel Vašata, Ph.D.

May 25, 2020

Acknowledgements

I would like to thank my supervisor, Ing. Daniel Vašata, Ph.D. for his advice and guidance throughout the process of developing and writing this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prag on May 25, 2020 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Koucký, Martin. *Analysis of discussion comments and their authors in social media.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstract

It is possible that among social media users there exist unknown clusters of users and anomalous users. This work explores that possibility by analyzing users represented by their comments.

We find suitable sources of data on social media sites and download them. Then, we propose vector representations of users based on their comments. Finally, we try to explain the clusters of users and anomalous users using various attributes on social media sites and with manual analysis.

Our results didn't prove the existence of clusters or anomalies among social media users, because there wasn't a clear distinction between normal and anomalous users or users of different clusters. This may have been caused by insufficient methods of representing users or manual analysis. But it may also mean, that there are no such clusters of users or anomalies commenting in a similar way to be found.

Keywords NLP, comments, users, anomalies, clustering, unsupervised learning

Abstrakt

Je možné, že na sociálních sítích existují shluky uživatelů nebo anomální uživatelé o kterých se neví. Tato práce tuto možnost prozkoumává tím, že analyzuje uživatele reprezentované jejich komentáři.

Našli jsme vhodné zdroje dat na sociálních sítích a stáhli z nich data. Poté navrhujeme matematické reprezentace uživatelů vytvořené na základě jejich komentářů. Nakonec se snažíme vysvětlit shluky uživatelů a anomální uživatele za pomocí atributů na socílních sítích a manuální analýzou.

Naše výsledky neprokázali existenci shluků nebo anomálií mezi uživateli sociálních sítí, protože jsme nenašli jasné oddělení normálních a anomálních uživatelů a uživatelů různých shluků. To mohlo být způsobeno nedostatečnými metodami reprezentace uživatelů nebo manuální analýzy. Mohlo by to ale také znamenat, že žádné shluky uživatelů nebo anomální uživatelé komentující podobným způsobem neexistují.

Klíčová slova NLP, komentáře, uživatelé, anomálie, shlukování, nesupervizované učení

Contents

List of Figures

Introduction

Most research about possible user clusters in social media has revolved around networks and connections between users. We focus on a different aspect of the problem. We want to determine whether there are any distinct clusters of users that comment similarly in online discussions. Identifying those clusters or anomalies would allow us to further analyze behaviour of those users and potentially classify them as anomalous users, trolls, or any other previously not thought of group.

To achieve this, we take the comments of uniquely identifiable social media users. Then we transform those comments into vector representations using state-of-the-art architecture BERT. Then we propose aggregations of those acquired vectors, because we need to represent each user as an individual data point. Finally, we apply K-means clustering and anomaly detection methods to the various user representations and analyze the results using social network attributes and manual analysis. We focus on finding distinct clusters of users and anomalous users as this is our main goal.

Chapter 1 explores previous work related to ours. Chapter 2 describes the methods we use in our experiments and Chapter 3 contains a description of our data sources, datasets, algorithms and analysis of the users.

Goals

Our goal in the theoretical part of the thesis is to research related works about vector representations of text data and clustering of users in social media. In particular, our research focused on modern approaches to representing sentences as vectors and finding anomalous clusters of users in discussions.

In the practical part of the thesis, our goal is to find suitable data sources and download enough data. Then use the researched NLP methods for vector representation of discussion comments and experiment with different parameters for user representation. After that, we use clustering algorithms on the users. Our final goal is to analyze the existence of clusters of social media users or anomalous users.

Chapter 1

Related work

In this chapter, we explore work related to ours. This gives us more insight and understanding of our problem as well as provides inspiration for new ideas we could use to solve our task.

1.1 Clustering of users in online discussions

Users on social networks interact with each other in various ways creating potential clusters inside the structure of the network in the process. Some of those clusters may be obvious such as friend networks on Facebook or followers on Twitter. Some are not as structured and require natural language processing approach or other machine learning methods.

Clustering users without explicit social ties has been the focus of work by Morrison et al. [1]. He found out that users could be clustered into four groups.

- **popular users** who contributed regularly and got a lot of replies
- **ignored users** who rarely got any replies
- **joining conversationalists** who communicated with very few other users
- **elitists** who communicated with high frequency but again with very few selected users

This partition has been done using nine features [2] describing mainly reciprocity, popularity and level of initialization of new threads. Meaning that the entire analysis was built without taking the content of posts and comments into account.

Krammer et al. [3] in their study *Clustering analysis of online discussion participants* tried adding information about content written by the users with

features `Vulg_count` and `Hoax_count`. Those features were created using a static bag of "vulgar words" and "hoax URLs" respectively and counting how many times they were in the user's posts.

Using those attributes they were able to identify similar structure of clusters across different datasets. The presented structure had one dominant cluster and neighbouring areas with lower density of data points. Those outlying points were classified as potential anomalous or extreme users. Their conclusion was that their approach could be used to identify anomalous behaviour in discussions.

1.2 Detection of anomalous users in online discussions

In their following research Krammer et al. [4] focused on finding methods for identifying anomalous users. They used the Kolmogorov-Smirnov test, which tests the hypothesis $H_0 : F = F_0$ that given data F correspond to a known distribution F_0 versus the hypothesis $H_A : F \neq F_0$ that they don't. The tested distribution was uniform which they believed to be in direct opposition to cluster tendencies in the data. From the low significance level of 0.05, they concluded, that the data should form clusters.

After that 4 main attributes were chosen to be used for clustering.

- **rel_react_to_me** representing average number of responses
- **rel_post_per_day** representing average number of posts per day
- **rel_word** representing post length in words
- **rel_violation** representing ratio of code violating posts / normal posts

Resulting clusters seemed to follow the standard distribution with one stable dominant cluster of "normal" users. The presented anomaly detection method was then to use canopy clustering multiple times and then classify data points according to a chosen threshold of significant cluster membership ratio.

1.2.1 Conclusion in relation to our work

This method can be used to classify some users as potentially unwanted and anomalous to some extent but doesn't provide any deep insight about potential multiple clusters in the data (only one cluster following standard distribution). The focus of our study is instead to try and find clusters that we didn't know previously existed at all.

1.3 Unsupervised detection of anomalous posts

In the past years, the phenomenon of spreading *fake news* has grown with the availability of the internet for everyone and the overflow of various information sources. There have been attempts to classify rumours in a supervised way, but Weiling Chen et al. in their study [5] took a different approach. They treated the posts containing false information as anomalies from the normal posts. Their reasoning was that most posts on the social media site would not be spreading false information, so the remaining small percentage of posts could be treated as anomalous.

To identify the anomalous rumours they use features extracted from the examined posts as well as features based on the corresponding user comments. They focus on user comments because previous research showed that rumorous posts are significantly more likely to be disputed by the commenting users. To take into account how these features vary over time, they use Recurrent Neural Networks. The features that are not time-dependent are then combined with these time-dependent ones and input into Autoencoder in order to detect anomalies.

1.3.1 Dataset

The authors used a dataset that comes from the Chinese microblogging website Weibo where posts/blogs are called *Weibos*. They collected n Weibos and then scraped each user u_i corresponding to some w_i from this dataset $\{w_0, ..., w_i, ..., w_n\}$ for his recent Weibos. In this way they acquired dataset $U = \{u_0, ..., u_i, ...u_n\}$, where $u_i = w_0, ..., w_k$. The number of extracted recent Weibos k can't be too large, because of the interference due to changes in users behaviour over time. However, it can't be too small either, because then the data wouldn't sufficiently model the user's behaviour.

To collect Weibos that are officially labeled as rumours, the authors collect data from *Weibo Community Management Center* where users send complaints about potentially rumorous posts and professional Weibo employees verify if they are in fact rumorous and make them visible for everyone. For the collection of the non-rumorous posts, the authors simply scrape the Weibo public timeline. Then they go through each collected post and manually verify it doesn't contain a rumour. Weibos obtained in this way at the start of the process form the original dataset of n labeled posts. From this dataset, each user is scraped for his recent comments as described in the former paragraph.

1.3.2 Features and their RNN transformation

Each collected post was described by a number of features concerning either its comments or the post itself. The post related features could be further divided in the following manner.

- **User interaction features** Counts of user comments, user likes and user reposts of the Weibo.
- **Post content features** Counts of pictures, # hashtags, @ mentions, smileys, first-person pronouns and question marks in the post and sentiment score of the Weibo.
- **Other post descriptors** Time when the Weibo was posted, whether the Weibo is a repost, source from where the Weibo was posted and length of the post.

The comment based features of the post could be divided as follows.

- **User interaction features** Count of likes of the comment, whether the commenter follows the original poster and vice versa.
- **Post content features** Counts of URLs, pictures, # hashtags, @ mentions, smileys, first person pronouns and question marks in the comment, the probability that the comment is positive and the probability that the author of the comment agrees with the post.
- **Other post descriptors** Whether the comment is a reply or a repost and the length of the comment.

Because the comment based features are time-dependent the authors use recurrent neural network (RNN) to capture their essence. They first divide the comments into a preset number of timeframes proportionally to the timestamp between the first and the last comment. The number of timeframes used in the paper is 7 as it can't be too high to avoid overfitting or too low so that it has enough information to learn behaviour patterns. To create suitable RNN input the comments that fall into the same timeframe need to be aggregated. To achieve this, the mean function is used in the paper. Then RNN is used to calculate feature interactions with respect to the comments time frames. The output of the RNN can be described as $W^O = (O_1, ..., O_7)$, where O_i is the i-th output generated from i-th hidden state.

1.3.3 Autoencoder and rumour detection

The outputs of the RNN, as well as the Weibo based features, are input into autoencoder which first transforms the input into a hidden layer with a smaller dimension. Then it transforms the hidden layer back into an output of the same dimension as the input (in the paper the input/output dimension is 119 and the dimension of the hidden layer is 50). The idea is that the autoencoder chooses the most important features of the data because it can only carry a limited amount of information through the "bottleneck" hidden layer with a lesser dimension.

Whether a Weibo_j of a user_i is a rumour or not is determined by calculating the reconstruction error $\text{Error}_{i,j}$ (the difference between input and output of the autoencoder) and comparing it to a threshold. This threshold is different for each user as each user has different behaviour, so to accurately detect rumours, each threshold_i has to be calculated individually for user_i based on his recent Weibos.

$$\text{threshold}_i = \text{md}_\text{i} + \max(1, \text{sd}_i)$$

Where md_i is the median of the reconstruction errors of the recent Weibos and sd_i is the standard deviation. Finally, the classification into rumorous or non-rumorous posts can be done.

$$\text{isRumour}_{i,j} = \begin{cases} 1, & \text{if Error}_i > \text{threshold}_i \\ 0, & \text{else} \end{cases} \tag{1.1}$$

Using these methods, the authors were able to achieve a relatively high accuracy score of 92.49% in the task of detecting rumorous posts in an unsupervised way.

1.3.4 Conclusion in relation to our work

We are not able to achieve similar results in part because our task isn't as clearly defined as classifying whether a comment is a rumour or not. And in part, because the data available to us aren't labeled. However the use of sentiment score as attribute in this paper provides support that the use of sentiment attributes in our user representations is meaningful.

1.4 Clustering of tweets

With the growing interest in twitter, navigating between the numerous tweets proved to be tedious. For this reason Kevin Dela Rosa et al. [6] studied the possibility of automatic clustering and classifying of tweets. The end goals were to cluster presented tweets into six predefined categories, cluster tweets into general topics, analysis of topic change over time and finding of the most representative tweet of a given cluster.

Their research showed that all the previous works in the same area used labelled or annotated datasets. They tried to come up with a more generalized way of annotating tweets with the use of Twitter's signature *hashtags*. Hashtags provide a user annotated description of the tweet in question with the use of keywords following a # symbol.

1.4.1 Corpus

To collect tweets that fell into the predefined categories, they analyzed trending news and chose the most used hashtags corresponding to the predefined categories. After that, a query to Twitter API was performed for each of those keywords every hour for two weeks resulting in a total of over one million tweets collected.

Table 1.1: An Example of the used categories and hashtags used to collect tweets from this category.

Category	**Hashtags used for collection**
News [#news]	#Japan, #wiunion, #obama, #libya, #gop
Entertainment [# entertainment]	#Radiohead, #ladygaga, #Bieber, #sheen, #Oscars

1.4.2 Unsupervised clustering

After tokenization and removal of rare terms, the authors wanted to determine how well would a standard clustering algorithm perform in the set task. They chose to use the TF-IDF weighting of tokenized words and measure K-means clustering against the predefined hashtags. They first calculated precision (P) and recall (R), which they used to calculate the pairwise F1 score ($F1$).

$$P = \frac{\mathrm{TP}}{\mathrm{TP+FP}}, R = \frac{\mathrm{TP}}{\mathrm{TP+FN}}, F1 = \frac{2PR}{P+R}$$

Where TP means true positive or confirmed positive, FP means false positive or classified positive that is actually negative and FN means false negative or positive that was classified as negative in its belonging to a hashtag-based cluster.

The achieved pairwise F1 scores were low for 30 different hashtag-based clusters (with a score of a 0.058) as well as for 6 different hashtags-based clusters (with score of 0.143) so the authors concluded, that tweets don't tend to naturally cluster according to their hashtags labels. For their task of classifying tweets into a predefined number of categories, this behaviour of unsupervised clustering was undesired so they proceeded to try supervised methods instead. With the use of the Rocchio classifier trained on hashtag labeled dataset, they were able to achieve much better results even in the 30 categories classification task (average score of 0.549 as opposed to the unsupervised score 0.058).

1.4.3 Cluster summarization

For a better understanding of the nature of discovered clusters, it is useful to find a human-readable representation of those clusters. One such representation can be the most "fitting" data point, in this case, a tweet. To find the N most representative tweets, the authors propose the following algorithm.

Construct centroid V of the given cluster C.
Initialize empty list T.
Sort tweets in C by their TF-IDF similarity with V.
for i in range $(0, N)$ **do**
  Pick highest ranked tweet t from C.
  if TF-IDF similarity of t to other tweets from $C \leq k$ **then**
    add t to T
    remove t from C
  end if
end for

Now the desired representative tweets are located in the T list. The threshold value k is set manually by the authors.

To evaluate how well would this algorithm perform, they calculated the precision score of the gained representative tweets based on human judgement evaluation with the use of multiple Amazon Mechanical Turk online workers to avoid inconsistency. The human workers chose whether the chosen tweet was highly relevant, relevant, or completely irrelevant. The final precision scores are much better for the algorithmically chosen tweets than for the randomly chosen ones. An important note is that the relevance of the tweets was being evaluated against a "story" query, which was a short sentence describing the cluster, created manually by the authors.

1.4.4 Conclusion in relation to our work

The authors utilized hashtags as a means of natural classification, which could be used to further represent comments and users in our work. Unfortunately hashtags aren't used enough on Twitter nowadays to provide consistent classification. The authors also tried unsupervised clustering and concluded, that tweets don't naturally cluster according to their original hashtag labels. However, it should be noted, that the sentence representations we use in our experiments are different than the ones that were used in this paper.

Chapter 2

Methods

In this chapter, we introduce and describe the methods used later in Chapter 3. The main part of this chapter explains how the natural language processing (NLP) model BERT and its parts work.

2.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) [7] is a relatively new natural language representation model developed as recently as in 2018. When a BERT model is pre-trained to represent plain text data with vectors, it can be used to represent text data in a number of NLP tasks. Many tasks that previously required long training can thus be performed only with some added fine-tuning of the output layer.

BERT is based on the concept of transformers [8]. The transformer is a new type of neural network architecture from Google. It proved to be both faster in training and achieve better results than the older recurrent neural networks (RNNs) or long short-term memory networks.

The recurrent models thus far had one big disadvantage in terms of speed. They couldn't be efficiently parallelized, because of the sequential nature of their computation. Transformers, on the other hand, completely disregard recurrence and rely completely on the concept of attention mechanism, which allows for parallelization.

2.1.1 Input embeddings

To generate text data input embeddings for the transformer, the sentence is first divided into tokens. Those tokens are then mapped to a vector space representing vocabulary of words according to the similarity of input words to words in the vocabulary thus generating *token embeddings*.

Another important part of the input embeddings are *positional embeddings* which indicate where in the sentence the given token lies. They must have the

same dimension as the *token embeddings*, because they need to be summed together later. They can be calculated in many ways, but BERT uses sine and cosine functions with different frequencies.

$$\mathrm{PE}_{(\mathrm{pos},2i)} = \sin(\mathrm{pos}/10000^{2i/d_{\mathrm{model}}})$$
$$\mathrm{PE}_{(\mathrm{pos},2i+1)} = \cos(\mathrm{pos}/10000^{2i/d_{\mathrm{model}}})$$

The final vector representation of an input token is calculated by summing the *token embeddings*, *positional embeddings* and *segmentation embeddings* that indicate which sentence the token belongs to.

2.1.2 Attention mechanism

The encoder in the transformer architecture consists of a multi-head attention layer and a feed-forward network layer. The attention layer tells us which part of the input should we focus on and depends on given input embedding. For example in the sentence "*This dog is brown and eats a banana.*" we should focus more of our attention on the word *dog* given the word *brown* than on the others, because of their stronger logical relation.

The encoder takes *Input embeddings* and creates key, values and queries from them. The keys and queries are of dimension d_k while the values are of dimension d_v. The values can be understood as all the words from the sentence. The keys correspond to sets of values that are related and "answer" the queries.

The Attention function first calculates the dot product of a set of queries (Q) against all the keys (K). Then it divides them by $\sqrt{d_k}$ to scale them. Finally, it applies a softmax, which assigns them a probability on a normal distribution. The output of the softmax function are the weights which are then used to multiply the original values (V).

$$\mathrm{Attention}(Q, V, K) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multiple-Head attention is a mechanism used to gain more varied information from the data. This is achieved by running h attention mechanisms in parallel. The keys, queries and values are first linearly projected h-times into a $d_{\mathrm{model}}/h = d_k = d_v$-dimensional matrices resulting in a total of h different matrices of types QW_Q, KW_K and QW_V, where $W_Q \in \mathbb{R}^{d_{\mathrm{model}} \times d_k}$, $W_K \in \mathbb{R}^{d_{\mathrm{model}} \times d_k}$ and $W_V \in \mathbb{R}^{d_{\mathrm{model}} \times d_v}$.

Then, the attention mechanism is applied h-times and the resulting matrices are concatenated together and projected by $W_O \in \mathbb{R}^{hd_v \times d_{\mathrm{model}}}$.

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concatenate}(\mathrm{head}_1, ..., \mathrm{head}_h)W_O,$$
$$\text{and } \mathrm{head}_i = \mathrm{Attention}(W_{Q,i}, W_{K,i}, W_{V,i})$$

2.1.3 Position-wise feed-forward neural network

The last component of the encoder is the fully connected feed-forward neural network (FNN). It is applied position-wise, meaning that each token vector x is run through an identical neural network using the same linear transformations. The process of applying FFN to x consists of rectifier activation function $max(0, xW_1 + b_1)$ and two linear transformations.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

2.1.4 Pre-training

One of the main ideas of BERT is to create a platform that can be easily used for a multitude of tasks with only a short fine-tuning procedure required at the start. This is achieved by the concept of pre-trained models which are obtained using two tasks.

BERT uses deep bidirectional model, because it better represents the meaning of a sentence. The bidirectional model couldn't be easily trained both left-to-right and right-to-left, because the model would indirectly see all the target words it was supposed to predict. Because of this, BERT uses Masked LM, which is in concept similar to the *cloze test*. The cloze test is an exercise, where the participant has to guess a word in place of a blank in a sentence. Following this concept, BERT randomly masks a small percentage of the tokens in an input and then predicts them.

To understand a relationship between sentences BERT pre-trains *next sentence prediction* on an unlabeled text corpus. When creating a train example of two sentences, it takes a sentence from the corpus and then chooses the next sentence to be either the actual following one (labeled *IsNext*) or random other one (labeled *NotNext*).

2.1.5 Pretrained models

In the original paper the authors used *BooksCorpus* with 800 million words and the *English Wikipedia* with 2,500 million words as pre-training corpus sources. Publicly available version[1] of this pre-trained BERT was thus only trained in understanding English texts. However, they later released BERT multilingual[2] which was trained on data from 100 Wikipedias of languages with the highest data volume. This pre-trained model included the Czech language as well.

The resulting model had a limited capacity for language understanding and language data volume varied across more and less used languages. Because of that, some languages would not be as well represented as others. To overcome

[1] https://github.com/google-research/bert

[2] https://github.com/google-research/bert/blob/master/multilingual.md

this, BERT developers weighted the data depending on the language during the training to compensate languages with less Wikipedia data.

The authors of BERT evaluate the performance of the multilingual model in the translation task on the Cross-lingual Natural Language Inference (XNLI) dataset [9]. XNLI provides dev and test pairs of text data translated by humans as well as machine-translated baselines for training and testing.

Table 2.1: BERT pre-trained models evaluation on XNLI dataset in the translation task

System	**English**	**Chinese**	**German**	**Urdu**
XNLI Baseline - Train	73.7	67.0	66.5	56.6
XNLI Baseline - Test	73.7	68.3	68.7	59.3
BERT - Train Cased	81.9	76.6	75.9	61.6
BERT - Train Uncased	81.4	74.2	75.2	61.7
BERT - Test Uncased	81.4	70.1	74.4	62.1
BERT - Zero Shot Uncased	81.4	63.8	70.5	58.3

Source: github.com/google-research/bert/blob/master/multilingual.md#results

Train The XNLI training set was first machine-translated *from* English into the desired language, then training and evaluation were both done in the desired language. The disadvantage is that we can't know what percentage of the error was caused by the machine translation and what percentage by the tested model.

Test The XNLI test set was first machine-translated *to* English from the desired language. Both training and evaluation were done on English. Again we can't know what percentage of the error was caused by the machine translation process.

Zero Shot In this case the BERT multilingual model was fine-tuned on English and then evaluated on XNLI test dataset of the desired language.

It should be noted, that even if those scores were low it wouldn't necessarily mean, that BERT is bad at representing one language. The only thing that would mean is, that either BERT doesn't learn language-independent properties of the languages or the languages the translation task test was performed on don't have a similar structure or other language property. Since the scores are fairly high and the former argument was examined and disproven [10], we can take that as at least some form of validation that pretrained multilingual BERT learns, to some degree, language-independent representations and thus can represent data in Czech as well.

Later, other BERT pre-trained models with different sizes, numbers of layers and numbers of hidden states were introduced[3].

[3]https://github.com/google-research/bert#pre-trained-models

2.2 LOF

Local outlier factor (LOF) was introduced by Markus M. Breunig et al. [11] in 2010. It is used to calculate and assign each object in a dataset a value describing its anomality (or, in terms of the original paper, outlierity). Compared to other outlier detection algorithms, LOF uses the concept of local density to find outliers in a local neighborhood. In practice, this means that for LOF an outlier can be even an object that seems to be in a cluster from a global perspective. However, the difference between how much of an outlier an object is compared to another is not lost, because LOF assigns an outlier factor instead of a binary of being or not being an outlier.

2.2.1 Formal definitions

Notations for objects that are used in the following definitions are: o, p, q. Notation for a distance between p and q is $d(p, q)$. C denotes a set of objects from a dataset denoted as D and $d(p, C)$ is the minimal distance between p and points from C, $d(p, C) = \min\{d(p, q) \mid q \in C\}$.

First, the k-distance of an object p has to be defined.

$$k\text{-distance}(p) = d(p, o), \text{ where } k \in \mathbb{N}, o \in D \text{ and satisfies:}$$

(i) for at least k objects $o' \in D \setminus p$ it is true, that $d(p, o') \leq d(p, o)$

(ii) for at most $k - 1$ objects $o' \in D \setminus p$ it is true, that $d(p, o') < d(p, o)$

Essentially, k-distance(p) is the distance between p and its k-th nearest neighbor. This distance is used in the following definition of k-distance neighborhood of an object p.

$$N_{k\text{-distance}(p)}(p) = \{q \in D \setminus p \mid d(p, q) \leq k\text{-distance}(p)\}$$

$N_{k\text{-distance}(p)}(p)$ contains k nearest neighbors of p, which are objects from D that are as close or closer than k-distance(p) to p. $N_{k\text{-distance}(p)}(p)$ is sometimes simplified to $N_k(p)$ when it doesn't cause confusion.

Reachability of an object p with relation to object o is defined as follows.

$$\text{reach-dist}_k(p, o) = \max\{k\text{-distance}(o), d(p, o)\}, \text{ where } k \in \mathbb{N}$$

reach-dist$_k(p, o)$ is introduced for smoothing the statistical fluctuations of $d(p, o)$, when p is close to o (in its neighborhood). The smoothing effect can be adjusted with different values of k.

Density is defined as mass divided by volume, and for the purpose of density-based algorithms such as [12] it is defined with MinPts and a parameter specifying volume, which are based on the original mass and volume. A density threshold is then calculated from those two parameters and examined

objects from a dataset are connected (form clusters) when the threshold is exceeded. In LOF it is necessary to calculate the density of objects dynamically. Because of that the volume parameter is calculated with reach-dist$_{\text{MinPts}}(p, o)$ for $o \in N_{\text{MinPts}}(p)$ and MinPts stays the same.

Local reachability density (LRD) of an object p is defined as.

$$\text{LRD}_{\text{MinPts}}(p) = 1/\left(\frac{\sum_{o \in N_{\text{MinPts}}(p)} \text{reach-dist}_{\text{MinPts}}(p,o)}{|N_{\text{MinPts}}(p)|}\right)$$

and finally local outlier factor of an object p is defined as.

$$\text{LOF}_{\text{MinPts}}(p) = \frac{\sum_{o \in N_{\text{MinPts}}(p)} \frac{\text{LRD}_{\text{MinPts}}(o)}{\text{LRD}_{\text{MinPts}}(p)}}{|N_{\text{MinPts}}(p)|}$$

This is the final output of LOF algorithm for one object and gives the information about how much of an outlier an examined object is. In the actual algorithm, LOF$_{\text{MinPts}}(o)$ is performed on each object $o \in D$ and thus all objects from a given dataset D have some factor/score of being an outlier assigned.

2.2.2 Properties of local outliers

The authors of LOF prove, that the following theorem is true for an object p from a dataset D if $1 \leq \text{MinPts} \leq |D|$ ($|D|$ is count of objects in D).

$$\frac{\text{direct}_{\min}(p)}{\text{indirect}_{\max}(p)} \leq \text{LOF} \leq \frac{\text{direct}_{\max}(p)}{\text{indirect}_{\min}(p)}, \text{ where}$$

(i) $\text{direct}_{\min}(p) = \min\{\text{reach-dist}(p, q) \mid q \in N_{\text{MinPts}}(p)\}$

(ii) $\text{direct}_{\max}(p) = \max\{\text{reach-dist}(p, q) \mid q \in N_{\text{MinPts}}(p)\}$

(iii) $\text{indirect}_{\min}(p) = \min\{\text{reach-dist}(q, o) \mid q \in N_{\text{MinPts}}(p), o \in N_{\text{MinPts}}(q)\}$

(iv) $\text{indirect}_{\max}(p) = \max\{\text{reach-dist}(q, o) \mid q \in N_{\text{MinPts}}(p), o \in N_{\text{MinPts}}(q)\}$

This theorem shows the general lower and upper bounds for LOF scores of an object of any type (outlier or inlier). The authors analyze how the tightness of the bounds changes depending on the ratio of direct/indirect and conclude that the theorem estimates the bounds well, if there is little fluctuation of average reachability distances in the dataset. This results in the effect that objects that are in the same cluster will have LOF scores close to 1, because the values of direct$_{\max}$, direct$_{\min}$, indirect$_{\max}$ and indirect$_{\min}$ are very similar.

Further, the authors analyze the impact of the chosen MinPts parameter on the resulting LOF score. When given an increasing sequence of MinPts, the corresponding maximum and minimum values of LOF scores do not change monotonically. Instead, the values fluctuate up and down and until they eventually stabilise when the value of MinPts is high enough. The authors

also observe that MinPts should be at least 10, because according to their experiments when the value was below 10, some objects had high LOF scores even though, they weren't outlying.

Finally, they propose a heuristic for determining LOF score of an object p : $\max\{\text{LOF}_{\text{MinPts}}(p) \mid \text{MinPtsLowerBound} \leq \text{MinPts} \leq \text{MinPtsUpperBound}\}$, where MinPtsUpperBound and MinPtsLowerBound are constants and have to be previously determined. In our experiments we used a similar heuristic: $\sum\{\text{LOF}_{\text{MinPts}}(p) \mid \text{MinPts} \in \text{PredefinedNumbers}\}$, where PredefinedNumbers are handpicked numbers from a range, in particular our chosen MinPts values in PredefinedNumbers were: 2, 4, 8, 16, 22, 42.

Chapter 3

Experiments

This chapter describes our experiments and their results. It details where from and how our datasets were obtained. Then we describe our representations and finally, we analyze users with those representations.

3.1 Data sources

In this section, we first analyze how sufficient are the different social sites as data sources. We need the source social site to have a clear identification of a user across his different comments and also enough users and traffic so that we can gather sufficient dataset. Then we describe how the data from those sites were scraped and the final datasets themselves.

3.1.1 Facebook

Following the Cambridge Analytica scandal, Facebook changed its policy regarding data scraping apps. For this reason, it is now impossible to scrape any meaningful data from Facebook without explicit user consent, which means that for our purposes the usage of Facebook as a source for data scraping is meaningless.

3.1.2 Twitter

We wanted to try analyzing Czech users as well as English speaking users. Because of that, we needed a social network with a sufficient Czech community. Additionally, we needed to be able to identify individual users. The only remaining social network satisfying those conditions with the Czech community besides Facebook was Twitter.

Twitter allows scraping of any user's data to anyone, but with strict limitations. Access to the Twitter API is divided into 15-minute intervals in which various forms of requests have their maximum limits set/reset. For example,

you can make 180 GET requests every 15 minutes. Twitter API also doesn't have any simple way of getting all comments from a single tweet. Let's say we have a **tweet** from **@Person** and we want all of its comments. But our tweet doesn't have any attribute identifying its comments. The only accessible feature of a tweet connecting it to another tweet is `in_reply_to_status_id` which tells us what tweet has our tweet been written in reply to. So the only way of getting comments for the given **tweet** is to use Twitter search and search for replies to user **@Person** and then take each found reply and compare the id of our **tweet** and the `in_reply_to_status_id` from the found reply. The search on twitter prioritizes newer tweets and older ones sometimes won't be found at all, which means that we can't get enough data if we just use this method for one scraping session.

Final datasets

To overcome the limitations, we started using Stream API which is continually searching for given tags (for example ["@Person1", "@Person2"]) and if a tweet is found and meets our requirements it is saved into a dataset of replies R. Because we don't just want comments alone, we now have to transform the dataset of replies to certain predefined people (usually famous Czech politicians, because they get the most replies) from R to a dataset that somehow represents users. To achieve this we take the names or ids of commenting users from R and scrape each one's profile for all of their comments (or some part of them) that match our requirements/restrictions. In our final datasets, we only accept tweets with the attribute `in_reply_to_status_id` not empty and only get comments/replies with a clear parent post. The final dataset U consists of a separate CSV file for each user with columns containing the text body of the users' tweet, the tweet he was replying to and information identifying both users. Although the origin of those users and their comments is a reply to a certain famous Czech person. The majority of those comments won't have any relation to that person in the end.

Additionally, in the period from 14 February to 5 April, we continuously scraped the Stream API for replies to a set of predefined Czech politicians – *@AndrejBabis*, *@PiratIvanBartos*, *@adamvojtechano*, *@alenaschillerov*, *@kalousekm* and *@tomio_cz*. We chose these politicians, because they were trending at the time of the scraping, so they would receive the most replies. Compared to the previously described dataset U, this dataset P contains for each user only replies to politicians from the predefined set.

The downloaded raw dataset contains 113,645 comments in total. Each comment has assigned category according to the parent user, which means that if the comment is a reply to @kalousekm, then the category is "@kalousekm". This classification is necessary, because some comments are delivered by the Stream API as replies only because they contain the "at sign" followed by a username of a politician from the predefined set, but in reality, they just

mention the politician and aren't direct replies. Those false positives make up 60% of the dataset. The two politicians with the most replies are *@adamvojtechano* with 16,514 comments and *@AndrejBabis* with 13,089 comments. The number of unique users corresponding to those comments are 2,190 and 1,454 respectively. Additionally, 16,893 of all the unique users in this dataset are not suitable for our experiments, because they only have one comment.

3.1.3 Reddit

Compared to the other two analyzed social networks Reddit offers unlimited access to user data, because users are almost completely anonymous. Reddit is divided by subreddits which are smaller independent message boards focused on some specific topic.

We decided to scrape users from specific subreddits, because this way we could better understand our data and find abnormalities. The users would all behave in a more similar way and thus any anomaly or cluster would be easier to find and more visible than if we scraped the highly varied Reddit users at random.

Final dataset

After we got the usernames from comments on posts in the subreddits we took each user and scraped his profile for a certain number of comments. Those comments weren't restricted to just one subreddit but they had to be replies – meaning they had a parent comment that they were reacting to. The final dataset contained one CSV file for each user and each row in that file contained attributes described in the table below.

Table 3.1: Reddit dataset columns/attributes structure

attribute name	**description**
child_body	text of the comment currently being scraped
child_screen_name	username of the user currently being scraped
child_user_id	ID of the user currently being scraped
child_comment_id	ID of the comment currently being scraped
parent_body	text of the parent comment
parent_comment_id	ID of the parent comment
parent_screen_name	username of the user who wrote the parent comment
parent_user_id	ID of the user who wrote the parent comment

We also scraped each user for his Reddit attributes `comment_karma`, `link_karma` and `comments_count`, which are described later. We saved them in a separate file, where each row represents one user, because we wanted to make the basic reddit and twitter datasets as similar as possible for ease of comparison in algorithms used later.

In our experiments, we mostly use dataset gained by scraping the subreddit /r/The_Donald for the supporters of Donald Trump. This subreddit is known to be controversial due to a high concentration of far-right leaning people and trolls among its members. We scraped its users for their comments to a maximum of 100 comments. In total, this dataset contains comments for 1,586 users.

3.2 User comments analysis

As our main goal is to determine the existence of user clusters based on their comments we first needed to find a suitable way to transform comments into vector representations. For this transformation, we decided to use BERT embeddings described in the section *BERT* of the Chapter *Methods*. We use the pretrained *Multilingual Cased BERT* [4] model to get the comment embeddings and comment sentiment[5] throughout all the experiments, because it supports the Czech language and is the most current multilingual version of BERT.

To get a basic idea about the structure of the data, we applied the K-means clustering algorithm to embedding representations of comments to tweets scraped from the twitter of the current Czech prime minister @AndrejBabis and to comments to submissions from the subreddit /r/AskReddit. Then we manually analyzed what meaning those clusters contain. From our analysis, it seems that the analyzed clusters that can be distinguished from others comprise of tweets with words that are close in topic or theme.

To determine whether differences between comments from those two sources can be recognized, we try to apply K-means with $k = 2$ to a combined dataset containing both of those datasets and compare the gained clusters with the true source of the data to see if the actual source corresponds to K-means created clusters. Only about 23 % of the comments were not clustered in the correct group according to their true dataset origin, which suggests that the differences between various groups of related comments are at least in some form recognized by the embeddings. It should be noted, that the main recognized difference was probably language (English for Reddit and Czech for Twitter).

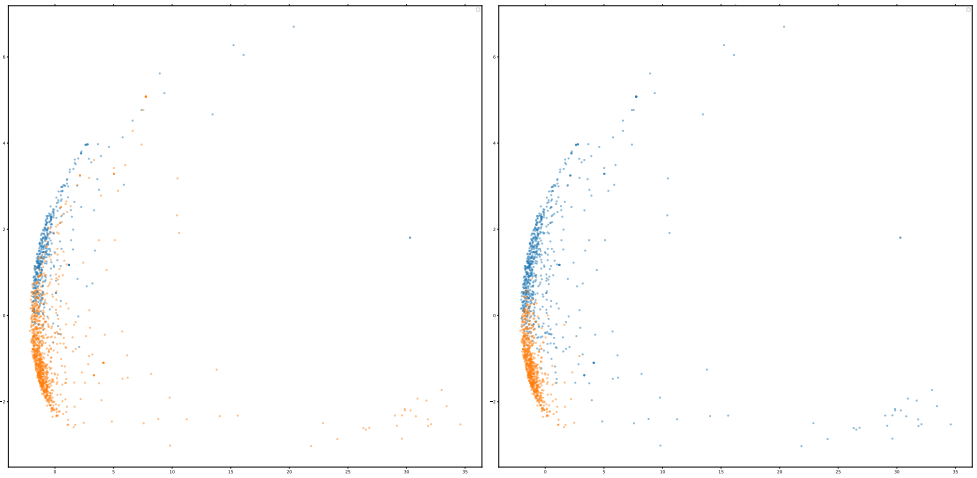
3.2.1 Results visualization

At first, we tried to use Principal Component Analysis (PCA) to reduce dimensions and visualize our results, but finally decided that the usage of t-distributed stochastic neighbor embedding (t-SNE) [13] is better for our data. PCA is not suitable, because it is a linear projection and doesn't handle reduction from such a high dimension to 2 dimensions well. Compared to that, t-SNE works well on the high dimensional, non-linear data from our dataset and represents the structure of manifolds and local neighbors relations better than PCA.

[4] bert_multi_cased_L-12_H-768_A-12

[5] sentiment is described in more detail in subsection 3.3.4

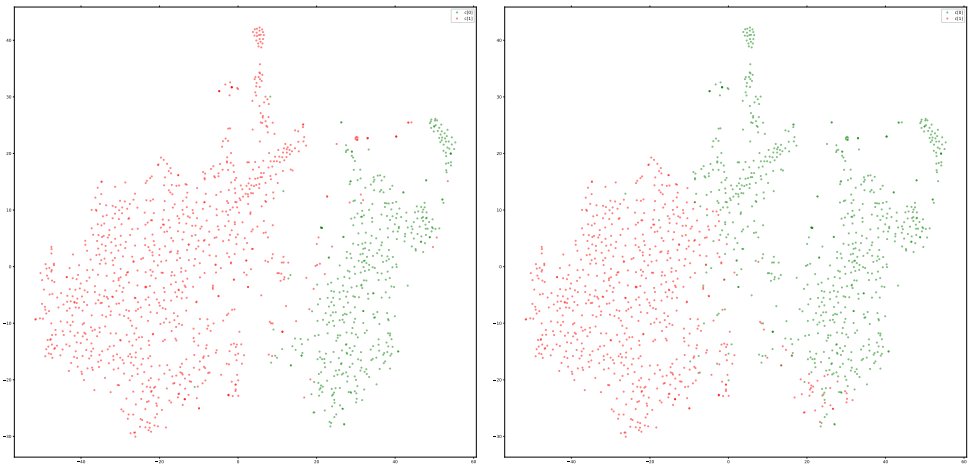
Figure 3.1: PCA visualization of comparison between /r/AskReddit comments and @AndrejBabis replies datasets



(a) actual/true dataset source; blue = /r/AskReddit, orange = @AndrejBabis

(b) K-means clusters, where $k = 2$

Figure 3.2: t-SNE visualization of comparison between /r/AskReddit comments and @AndrejBabis replies datasets



(a) actual/true dataset source; green = /r/AskReddit, red = @AndrejBabis

(b) K-means clusters, where $k = 2$

3.3 User representations

To sufficiently represent each user as one data point we had to aggregate his comments in a meaningful way to gain as much information from the combined comments as possible. For a given dataset $D_U = \{U_1, ..., U_k\}$ each user matrix $U_i \in D_U$ contains n_i rows of comment embeddings, where $1 < n_i <= n$. This means that we scraped users for a maximum of n comments and dismissed users with less then 2 comments, because later we use methods that require at least 2 embedding vectors. Each row / comment embedding has 768 columns/dimensions representing the original comment in a vector space.

Additionally, there is the dataset $D_P = \{P_1, ..., P_k\}$ of parent comment embeddings, where each matrix P_i contains comments[6] to which the comments in U_i were replies. Our dataset only contains replies to a parent comment, so each row from $U_i \in D_U$ has a corresponding row in $P_i \in D_P$.

3.3.1 Mean of comment embeddings

With this approach we simply calculate the mean of each column/dimension of a row/vector for each user matrix U_i. This gives us new aggregated row M_i representing user. All the rows are then added together to create matrix M representing the means of comment embeddings for each user in the dataset D_U.

$$M = \text{EmbeddingMeans}(D_U) = \text{concatenate}(M_1, ..., M_i, ..., M_k), \text{ where}$$
$$M_i = (\text{mean}(U_{i,0}), ..., \text{mean}(U_{i,j}), ..., \text{mean}(U_{i,768})) \text{ and}$$
$$U_{i,j} \text{ is the } j\text{-th column of } i\text{-th user}$$

The resulting matrix M has k rows/users and 768 columns and provides us the most basic aggregation of the information contained within BERT embeddings. However, this simple approach might be prone to error while using datasets with large n and with users that engage in a variety of topics, because in both of these cases the mean function normalizes any present information.

3.3.2 Relativization of comment embeddings

To gain information about the style in which the user responds to other comments, we try to relativize user comments. For each user, we take his comment embeddings matrix U_i and subtract corresponding parent comment embedding matrix P_i creating matrix R_i in the process. All the relative user representations make up new dataset $D_R = \{R_1, ..., R_k\}$. After that, the EmbeddingMeans function is applied and we gain the relativized comment embeddings matrix R.

[6] *comments* means comment embeddings here, simplified *comments* is used for clarity of understanding the origin

$$R = \text{EmbeddingMeans}(D_R), \text{ where}$$
$$D_R = \text{relativize}(D_U, D_P) = \{R_1, ..., R_i, ..., R_k\}, \text{ and}$$
$$R_i = U_i - P_i$$

The previous problem with too much variety in topics shouldn't be as substantial, because the user should, in theory, respond in a similar way to most topics.

3.3.3 Variance of comment embeddings

To account for the diversity of topics the user has engaged in, we propose the usage of variance. We work with U_i matrices from the D_U dataset and calculate the variance of each column of U_i. This results in a row of variances V_i which represents one user. Those rows are added together to create the matrix V representing variance in topics and meaning of comments for each user.

$$V = \text{EmbeddingVariance}(D_U) = \text{concatenate}(V_1, ..., V_i, ..., V_k), \text{ where}$$
$$V_i = (\text{var}(U_{i,0}), ..., \text{var}(U_{i,j}), ..., \text{var}(U_{i,768})) \text{ and}$$
$$U_{i,j} \text{ is the } j\text{-th column of } i\text{-th user}$$

3.3.4 Sentiment analysis

We trained modified BERT example classifier[7] on the *Large Movie Review Dataset v1.0* [14], which was then used to predict whether the sentiments of given comments from D_U and corresponding D_P were negative (0) or positive (1). For user of index i, the newly created sentiment matrix S_i contained 6 columns/attributes described in table Table 3.2 below and n_i (count of comments) rows.

These attributes give us information not only about sentiment, but also about the relation and rate of potential agreement or disagreement of the user in question with the comment he was replying to. The gained matrices S_i are then concatenated thus creating the S matrix representation of users.

$$S = \text{concatenate}(S_1, ..., S_i, ..., S_k),$$

$$S_i = (\text{mean}(\text{sentiment}(U_i)), \text{mean}(\text{sentiment}(P_i)),$$
$$\text{Ratio}_{00}, \text{Ratio}_{01}, \text{Ratio}_{10}, \text{Ratio}_{11}), \text{where}$$

$$\text{Ratio}_{a,b} = \frac{\sum_{j \in n_i} [\text{sentiment}(U_{i,j})=a \text{ and } \text{sentiment}(P_{i,j})=b]}{\sum_{j \in n_i}},$$

for ratios 00, 01, 10 and 11 and
U_i, j and P_i, j is the j-th row of i-th user for datasets D_U and D_P respectively.

[7] github.com/google-research/bert/predicting_movie_reviews_with_bert_on_tf_hub

Table 3.2: Sentiment attributes in S_i

Attribute	Explanation
Comment sentiment	Mean of sentiment of the user comments from U_i [either 0 or 1]
Parent sentiment	Mean of sentiment of the parent comments from P_i [either 0 or 1]
00	Ratio of both user comment and parent comment being negative
01	Ratio of user comment being negative and parent comment positive
10	Ratio of user comment being positive and parent comment negative
11	Ratio of both user comment and parent comment being positive

Because we couldn't use the English IMDB dataset to train the classifier for our Czech comments from Twitter, we used the *Facebook CZ* [15] labeled dataset instead. In this dataset, every comment is labeled as positive, neutral, negative or bipolar. Because this didn't fit into our original structure of sentiment analysis (either negative or positive), we created two new sentiment training sets. One with the same structure as the original, disregarding bipolar and neutral answers. The other would train the classifier to predict how emotional the user was by relabeling negative, positive and bipolar to "emotional" (or 1/positive) and neutral to "indifferent" (or 0/negative).

3.3.5 Other representations

We also tried to represent users with attributes not based on the text of comments alone like users karma or count of comments on Reddit, but ultimately decided against it, because we don't want the clusters to be based on those simple arguments and because the inclusion didn't seem to improve the existing clustering. Additionally, because we don't use those attributes for clustering, we can use them to explain user clusters later.

3.4 User analysis

We use three main combinations of user representation methods from the previous section to describe users. Simple mean (**M**), mean in combination with variance (**MV**) and relativized embeddings in combination with variance (**RV**). The first is used as a benchmark and the simplest form of user description upon which the others are based. The second combination with added variance is more theme focused and describes what range of topics the user engages in and the last describes the style (due to relativization) in which the user responds to a topic (due to variance). We also tried adding the sentiment attributes from previous section to each combination (**SM**, **SMV** and **SRV** respectively). In this section, we are going to use various tools to analyze users represented by these combinations.

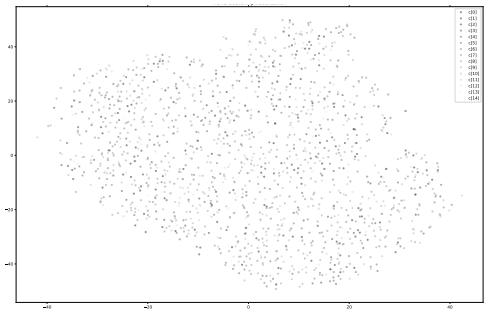
3.4.1 Explanation via correlation with Reddit attributes

Our Reddit datasets contain, in addition to standard comment related data, information about users comment karma, post karma and count of all comments. Comment karma is the sum of upvotes and downvotes on all of the users' comments. Post karma, similarly, the sum of upvotes and downvotes on all of his posts. The equivalent on Twitter would be an original tweet for post and reply for comment.

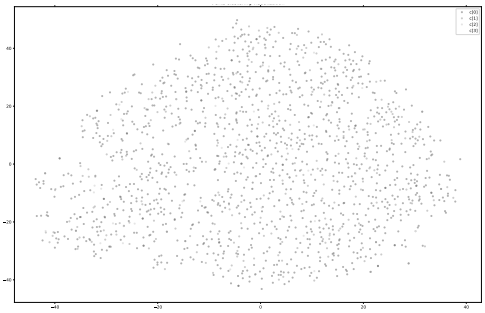
We use the K-means algorithm to cluster our user representations and then try to see abnormalities or differences in their Reddit attribute values. To simplify those Reddit attributes for comparison purposes, we first calculate the logarithm of each attribute, which should give us a better idea about the relative difference between users, because there isn't as much difference between a user with a karma of 1500 and a user with a karma of 1700 as there is between users with -100 karma and 100 karma respectively. For negative values logarithm is calculated as $-\log(|\text{negative value}|)$. Then we calculate the mean of all attributes in a given cluster to represent the cluster as a whole.

Further, we try to explain the relationship between Reddit attribute values and our representations of users by plotting them in 2D representation with the help of t-SNE and differentiating various logarithmic attribute values with a color gradient. In this way, we could find possible patterns of increasing or decreasing values of Reddit attributes.

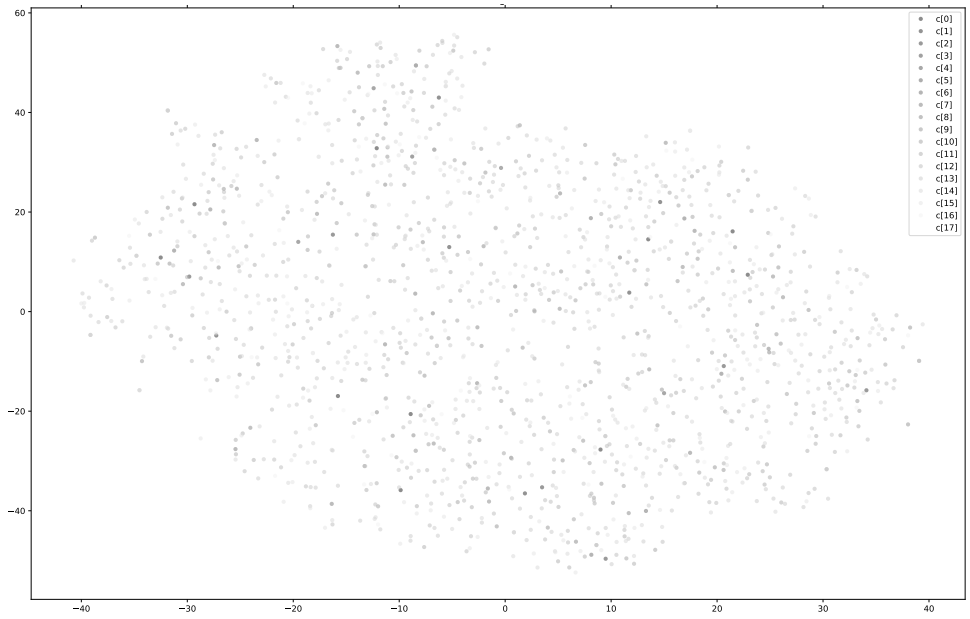
Figure 3.3: t-SNE visualization of users Reddit attribute values by color gradient on /r/The_Donald users dataset represented by the **RV** combination (c[i] are ascending log values of Reddit attributes)



(a) post_karma attribute



(b) comment_count attribute



(c) comment_karma attribute

Unfortunately, none of the K-means clusters seems to have distinctly different mean values of any of the attributes and color gradient plot didn't show any relation between values and user representations either. We conclude that either there isn't a relation to be found and karma and count of comments don't correlate with differentiation by comments. Or that we haven't tested on sufficient data and the small differences between clusters would be more significant and cleaner on larger datasets.

When we visualized the dataset with users from Twitter replying to Czech politicians, we noticed clear and distinct clusters. Unfortunately, further analysis indicates that those clusters appear, because the Czech Twitter dataset

consists in a large part of users with small amounts of comments (a majority of users from this dataset are represented only by 2 comments). This causes the clusters to appear when the sentiment attributes are added, because for the users with small amounts of comments, there isn't enough variety in sentiment combinations to differentiate them. We visualize this problem with the gradient rounded sentiment and rounded sentiment plots (C.1, C.2) for the **SRV** representation and compare them to the same types of plots for users represented only with sentiment attributes (C.3, C.4) to show their relation.

3.4.2 Anomalous users

For the detection of anomalous users, we use Local Outlier Factor (LOF) and modified K-means algorithms. The gained anomalies weren't distinct from normal users when we tried explaining them with their attributes (Reddit attributes and mean values of sentiment). Because of that we manually analyze comments of users classified as the most anomalous and try to find any similarities between them or distinctions from normal users.

LOF anomalies

We analyze users with the highest LOF scores from all the representation combinations (**M**, **MV**, **RV**, **SM**, **SMV**, **SRV**) for a total of 6 sets of users. Our manual analysis consisted of reading and examining a number of the newest comments that the user in question made. The number of comments we examine for each user ranges from about 20 to 100 depending on the length and complexity of the comments. We focus on the meaning, style, length and sentiment in the comment. Secondary we look at how well it is written and if any swearing occurs. We also try to take other available attributes like where the comment was posted and the number of likes/upvotes into account.

We first perform our analysis on the dataset with commenters from */r/The_Donald* subreddit. Some of the most anomalous users represented by **M** and **SM** have posted multiple "spam" comments of the same content as a reply to different parent comments. This is unique to those representations (or rather they aren't classified as the most anomalous in other representations). Users from **MV**, **RV**, **SMV**, **SRV** show signs that could be interpreted as anomalous like swearing, banned comments or one-word comments, but none of those signs are consistent across majority of examined users. Some of the anomalous users also have new accounts and not many comments, which could have caused them to be classified as anomalous just because their representations haven't been normalized as much due to the usage of mean function in the creation of user representations.

We also perform our analysis on the dataset with users replying to Czech politicians on twitter. We analyze 6 most anomalous users of each representation. We examine additional attributes, because Twitter offers unique features such as the number of followed accounts, number of followers and user description. Some of the examined users tweet in another language and even reply in that language to Czech threads, this may be the cause of their classification as anomalous. Notably, one of the 6 most anomalous users of the **RV** representation was a well known Czech journalist *@jindrichsidlo* and one of the anomalies of the **SM** representation was a banned account. The majority of the anomalous users reply mostly negatively or ironically, but that seems to be the case for all users in political debates on Twitter.

K-means anomalies

We detect anomalies with the usage of the K-means algorithm with minor modifications. First, we create an array containing all users. Then we run K-means multiple times with different numbers of clusters k. After each clustering, the value of total distance for each user is updated by adding the distance to its current corresponding cluster centroid. Users in the array are sorted by their total distance, which gives us the most and the least anomalous users while taking into consideration different sizes of clusters. We also tried sorting the users each time they were clustered and remembering the most anomalous users of each iteration, then sorting them by the number of times they were among the most anomalous, but this brought the same results (the same set of anomalous users) as sorting by total distance and didn't give as much information.

We perform our manual analysis on the same 6 representation combinations as we do with LOF and on the same dataset of */r/The_Donald* commenters. Again some of the most anomalous users reply with "spam" comments, but this time they are present in all the representations except for **RV**. Even though there are different users, the same trends as described in LOF anomalies are present. Some users have new accounts, some swear, some don't, some have a lot of downvotes or upvotes and some have comments with no engagement from other users at all. Majority of users seem to comment primarily on */r/The_Donald* and related (political) subreddits, but there are exceptions.

Because there doesn't seem to be a clear defining characteristic of users classified as anomalous, we try to analyze the least anomalous and compare them. Majority of them don't swear and their comments aren't hateful, but again that could be subjective and there are also exceptions, meaning that we cant say that they are the most "normal" in reality. None of the most "normal" users appear to have new accounts or not enough comments. This leads us to believe that the normality could just be caused by a larger, more varied set of comments which is then normalized by the mean function.

As was the case with LOF, we perform additional analysis on the dataset with users from twitter replying to Czech politicians. When we analyzed LOF anomalies the top anomalous users were different for each representation, this wasn't the case with K-means. Only a few of the top anomalous users change across all the representations. This is interesting, because in the Reddit */r/The_Donald* dataset the LOF and K-means anomalies were similar in the presence of the same top anomalous users across the different representations.

Notably a user with suspended account was classified as anomalous in the **MV**, **RV**, **SMV** and **SRV** representations. Most tweets of one of the users that was present in top anomalous users in all the representations are retweets or links to other sites and videos. Recurring is also a foreign-speaking user that comments in Czech presumably only thanks to Google translator. Two of

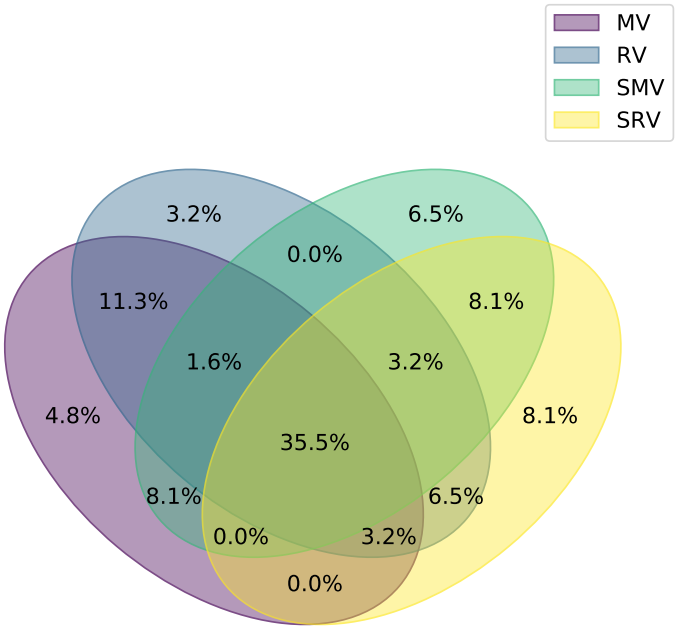
the recurring users use swear words frequently and appear hateful. But there are also normal users that don't present anything that could be classified as anomalous.

Conclusion

We tried to find out if the users classified as anomalous by LOF and K-means algorithms were, in fact, anomalous or shared a common trait with the use of manual (human) analysis. Behaviour that could be described as anomalous could be observed in some of the outliers, but some of the top outliers didn't present any abnormal behaviour at all. It should be noted, that outliers don't have to be necessarily abnormal in the same way or have anything in common. Because of this diversity, which is hard to observe, it is hard to determine whether our outliers still have some value to them.

Many of the users are present in more than one group of classified top anomalies on a specific representation. We visualized this with the use of Venn diagrams to better understand it.

Figure 3.4: Venn diagram visualization of intersections between groups of top 40 anomalous users classified with K-means represented by the **MV**, **RV**, **SMV** and **SRV** combinations.



Conclusion

The goal of this thesis was to research existing work and methods related to NLP and analysis of users and anomalies in an unsupervised way and then download sufficient data and perform our own analysis.

We downloaded unlabeled comment data from Twitter and Reddit and using BERT transformed them into vector representations. Then we proposed forms of aggregated representations of users from the comments and applied K-means clustering and anomaly detection methods on the representations.

We tried to explain the gained clusters and anomalies with their relation to various attributes, but they didn't correlate in a significant and consistent way. Then we tried to explain the users classified as anomalies with manual analysis. Although some portion of the most anomalous users presented behaviour, that could be classified as anomalous by a human observer, not all of them did. Further, we examined the users that were not classified as anomalous and some of them showed the same behaviour patterns as the anomalous ones. None of the behaviour patterns we examined was consistent across a significant enough portion of the anomalies.

This may be caused either by the loss of some information and the overflow of unimportant information in our proposed user representations. Or because there aren't clusters of users that comment in a similar way or anomalous users distinct from the normal ones to be found. This may be solved by only considering some elements of the text data and taking time into account in the future.

Bibliography

[1] D. Morrison, I. McLoughlin, A. Hogan, and C. Hayes, "Evolutionary clustering and analysis of user behaviour in online forums", in *ICWSM*, Jan. 2012, pp. 519–522. [Online]. Available: https://www.researchgate.net/publication/290559734_Evolutionary_clustering_and_analysis_of_user_behaviour_in_online_forums.

[2] J. Chan, C. Hayes, and E. M. Daly, "Decomposing discussion forums and boards using user roles", in *ICWSM*, 2010. [Online]. Available: https://www.semanticscholar.org/paper/Decomposing-Discussion-Forums-and-Boards-Using-User-Chan-Hayes/a09b8ae46c58563d100b09794d6ab1c802e23f59.

[3] P. Krammer, M. Kvassay, J. Mojžiš, I. Budinská, L. Hluchý, and M. Jurkovič, "Clustering analysis of online discussion participants", *Procedia Computer Science*, vol. 134, pp. 186–195, 2018, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.07.161. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050918311256.

[4] P. Krammer, O. Habala, J. Mojžiš, L. Hluchý, and M. Jurkovič, "Anomaly detection method for online discussion", *Procedia Computer Science*, vol. 155, pp. 311–318, 2019, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.08.045. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050919309597.

[5] W. Chen, Y. Zhang, C. K. Yeo, C. T. Lau, and B. S. Lee, "Unsupervised rumor detection based on users' behaviors using neural networks", *Pattern Recognition Letters*, vol. 105, pp. 226–233, 2018, Machine Learning and Applications in Artificial Intelligence, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2017.10.014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865517303823.

[6] K. Dela Rosa, R. Shah, B. Lin, A. Gershman, and R. Frederking, "Topical clustering of tweets", *3Rd Workshop on Social Web Search and Mining*, Jan. 2011. [Online]. Available: https://www.researchgate.net/publication/267805712_Topical_Clustering_of_Tweets.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. arXiv: 1810.04805 [cs.CL].

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[9] A. Conneau, G. Lample, R. Rinott, A. Williams, S. R. Bowman, H. Schwenk, and V. Stoyanov, *Xnli: Evaluating cross-lingual sentence representations*, 2018. arXiv: 1809.05053 [cs.CL].

[10] I. Kvapilíková, "Unsupervised machine translation between czech and german language", Czech Technical University in Prague, Faculty of Information Technology, 2020.

[11] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander, "Lof: Identifying density-based local outliers.", vol. 29, Jun. 2000, pp. 93–104. DOI: 10.1145/342009.335388.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96, Portland, Oregon: AAAI Press, 1996, pp. 226–231. [Online]. Available: https://dl.acm.org/doi/10.5555/3001460.3001507.

[13] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE", *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[14] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis", in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA: Association for Computational Linguistics, 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015.

[15] I. Habernal, T. Ptáček, and J. Steinberger, "Sentiment analysis in czech social media using supervised machine learning", in *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Atlanta, Georgia: Association for Computational Linguistics, 2013, pp. 65–74. [Online]. Available: http://www.aclweb.org/anthology/W13-1609.

Appendix A

List of used abbreviations

NLP Natural language processing

RNN Recurrent neural network

FNN Feedforward neural network

PCA Principal component analysis

t-SNE t-distributed stochastic neighbor embedding

LOF Local outlier factor

Appendix B

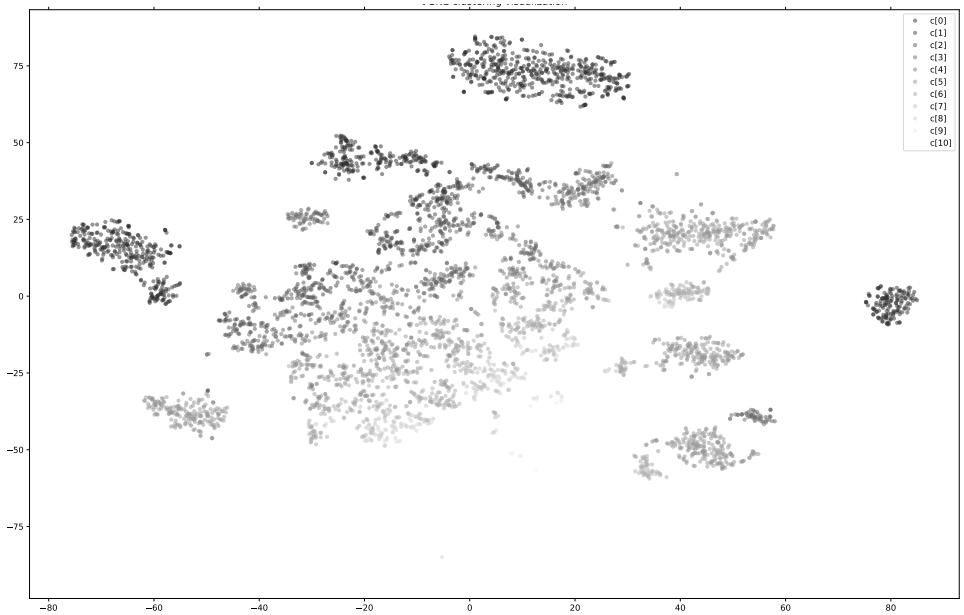
Contents of the enclosed USB

- `readme.txt` ..... short description of the contents of the USB
- `web_scraping` ..... webscraping source codes and raw downloaded data
  - `facebook` ..... unsucesfull Facebook scraping attempt
  - `reddit` ..... Reddit webscraping sources codes and data
  - `twitter` ..... Twitter webscraping sources codes and data
- `downloaded_datasets` . folder containing structured downloaded datasets
- `src` ..... GoogleColab source codes for experiments
- `representations` ..... final user and comment represention in CSV files
- `plots` ..... plots from experiments in the form of PDF
- `text` ..... text of the thesis in the form of PDF

Appendix C

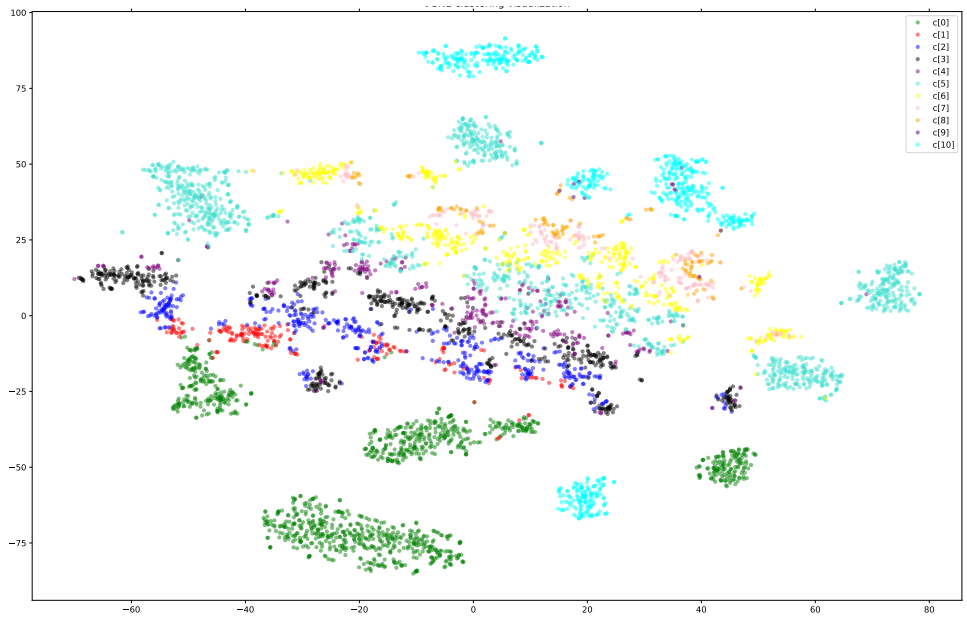
Plots

Figure C.1: t-SNE visualization of user comment sentiment on the Twitter Czech politicians dataset using **SRV** representations



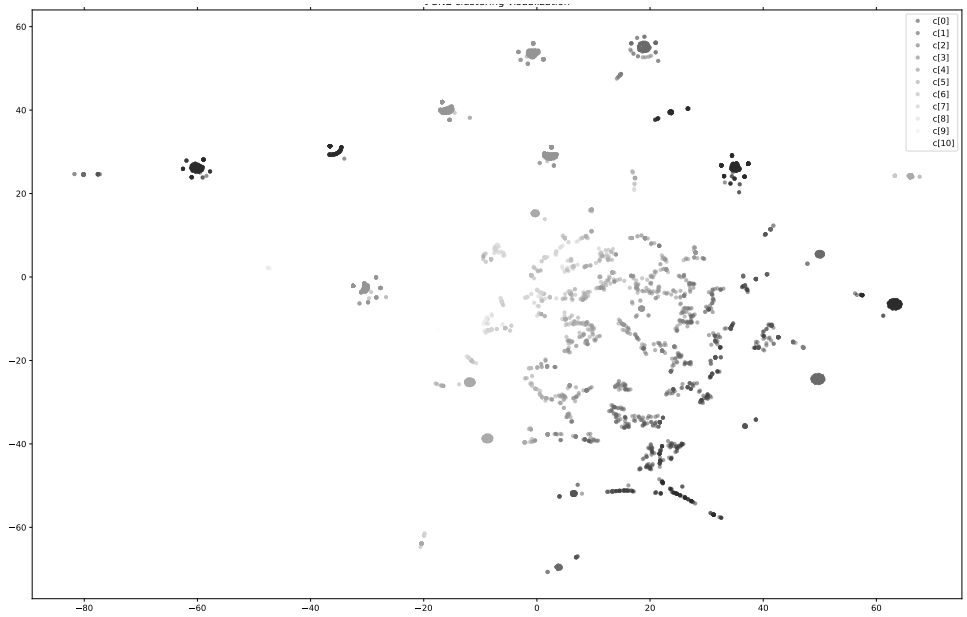
the lighter the shade of gray – the higher (more positive) the average value for sentiment of user comment

Figure C.2: t-SNE visualization of user comment sentiment on the Twitter Czech politicians dataset using **SRV** representations



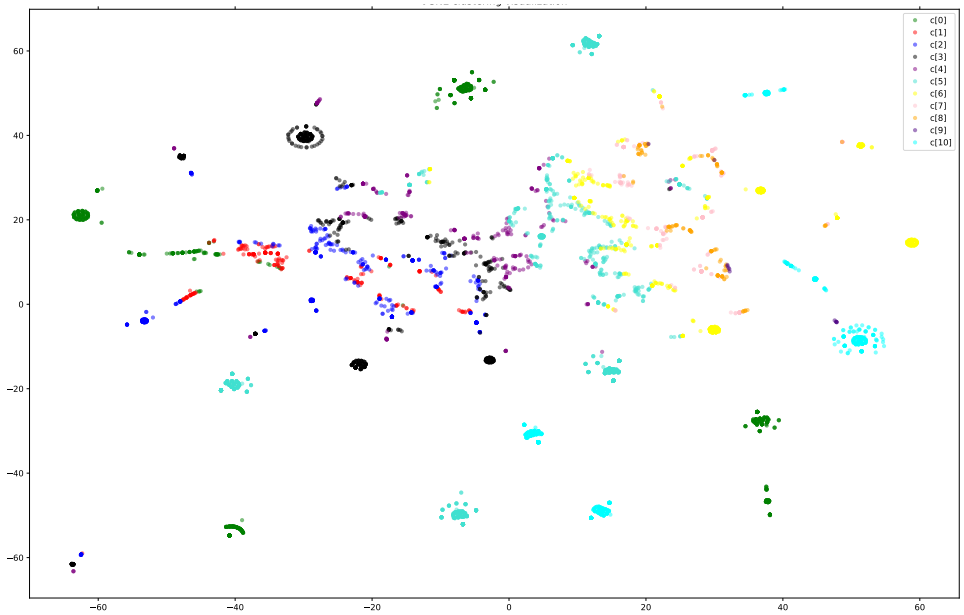
each color/c[i] denotes rounded average value of user comment sentiment,
for example c[0] = users with average sentiment around 0–0.05 (negative)

Figure C.3: t-SNE visualization of user comment sentiment on the Twitter Czech politicians dataset using only sentiment attributes as a representation



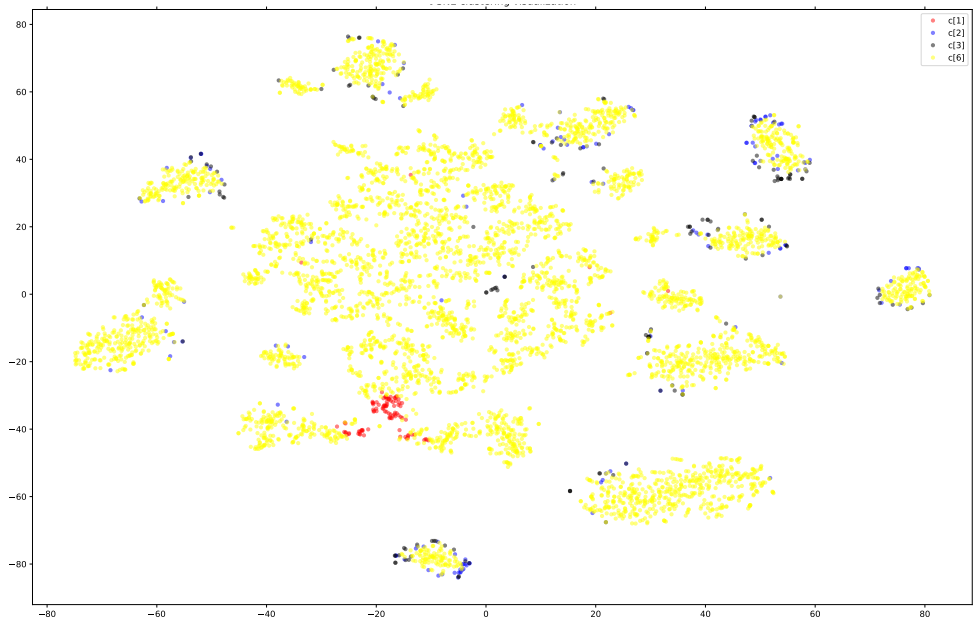
the lighter the shade of gray – the higher (more positive) the average value for sentiment of user comment

Figure C.4: t-SNE visualization of user comment sentiment on the Twitter Czech politicians dataset using only sentiment attributes as a representation



each color/c[i] denotes rounded average value of user comment sentiment, for example c[0] = users with average sentiment around 0–0.05 (negative)

Figure C.5: Example of K-means anomalies on the Twitter Czech politicians dataset using **SRV** representation



[black] = 150 most anomalous users, [blue] = 150 second most anomalous users, [yellow] = normal/other users, [red] = 100 least anomalous users