ELEKTROTEHNIŠKI VESTNIK 81(4): 160–166, 2014
ORIGINAL SCIENTIFIC PAPER

Movie sentiment analysis based on public tweets

Aljaž Blatnik[1], **Kaja Jarm**[1], **Marko Meža**[1]

[1]*Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia*

† *E-mail: aljaz.blatnik@gmail.com, kaja.jarm@gmail.com*

Abstract. In this paper we present a different approach to understanding the natural language and frequency of words from one sentiment class enabling a better sentiment classification compared to the traditional machine learning techniques. We also introduce one more sentiment class - the neutral class. In our research we use the Python programming language with the NLTK library and compare thus obtained results with the traditional machine learning techniques using RapidMiner. Our focus is on Twitter - a microblogging platform with a maximum of 140 characters per post (tweet), more specifically, on gathering the sentiment for certain movies.

Keywords: Twitter, sentiment analysis, sentiment classification, opinion mining, Twitter data analysis, text mining, machine learning, big data

Zaznavanje javnega mnenja o filmih na podlagi sporočil Twitter

V okviru tega članka želimo predstaviti drugačen način klasifikacije mnenja v okviru razumevanja naravnega jezika in frekvence besed v posameznem razredu mnenja v smislu boljše klasifikacije kot s tradicionalnimi metodami strojnega učenja. Predstaviti želimo tudi možnost vpeljave dodatnega nevtralnega razreda mnenja. Za analizo smo uporabili programski jezik Python z dodatno knjižnico NLTK. Rezultate analize z našim algoritmom smo nato primerjali z rezultati tradicionalnih metod strojnega učenja v programu RapidMiner. Za raziskavo zastavljene teme smo se osredinili na socialno omrežje Twitter, ki je t. i. koncept mikroblogov z omejitvijo 140 znakov na posamezen objavljen zapis. Specifično področje, ki smo ga raziskovali, je javno mnenje o posameznih filmih.

1 INTRODUCTION

Presently, one of the widely used aspects of the web are social networks, one of which is Twitter. Its first defining feature are 140 characters per post (or as Twitter calls it "tweet"). This limits the user to compose a relatively short message with the content to be expressed. The second defining feature is the use of the hashtag symbol. People use the hashtag symbol # before a relevant keyword or phrase (no spacing) in their tweet to categorize those tweets and to mark keywords or topics in a tweet [1]. Examples of the hashtag are: #movie, #wolfofwallstreet, etc. There are three phrases motivating the use of Twitter listed on the Twitter discover page [2]: "Connect with people, Express yourself, Discover what's happening". Those three phrases and the two defining features altogether offer a lot of possibilities for the data consisting of 500 million tweets posted daily by 241 million monthly active users [3]. With that amount of data, organizations of any kind can get the feedback on their product without even bothering their customers with surveys. All they have to do is first find the relevant tweets and then analyze them.

With so many accessible data on Twitter and the movie-rating systems being quite complex (logging into an application or a web page and then rating or even reviewing the movie), our aim research was to simplify the current movie-rating systems. Using 140 characters (or even less) the Twitter users post on their profiles we wanted to make a decision system to detect whether the public opinion on a certain movie is good, bad or something in-between. The opinion categories we used were: positive, neutral and negative. This is also known as sentiment analysis. Although the Twitter users are mostly average viewers, not the demanding ones, our research was able to make a good representation of the general public opinion about a movie by extracting the opinion on a large group of people.

Analysis of the social networks is currently widely performed in the research field. Research efforts are much spread. Authors [5] claim that Twitter functions as an electronic word of mouth at the USA Pennsylvania State University. Several studies focus on sentiment analysis within social networks. For example, authors of [6], studied how the public sentiment changes during events. Other studies solve problems like ours ([10], [8]),

Received 29 May 2014
Accepted 24 June 2014

but they focus on more than one field (for example they study the sentiment in relation to movies, food, brands, etc.). Our decision was to focus on one single field, the public sentiment about the currently popular movies.

There are various approaches to analyzing tweets, but in our research we analyzed the natural language with the text-processing algorithms already implemented in some of the data-mining programs such as RapidMiner. Our main hypothesis was that a) using simple statistical approaches and b) understanding the natural language will provide better results on a small training dataset than widely used algorithms for data mining. To prove that, we developed a new tweet-classification algorithm.

Another aspect that makes our research different from other researchers is the way we acquired the data. Compared with other papers, for example [8], our data acquisition is completely manually processed opposing to the automatic one based on emoticons with added keywords. One of the reasons why we believed our approach would work better compared with the others was us using a highly precise training dataset. The other reason was the feature we added, i.e. the neutral class. Unlike other researchers, we introduced a neutral sentiment class for the tweets which are neither positive nor negative.

2 MATERIALS AND METHODS USED

2.1 Data acquisition

In order to build a real-data model, a reliable dataset was needed. After observing the available data corpus of tweets [8], we decided to manually build our own training and testing dataset since: 1.) most of the datasets available were automatically built based on the detected emoticon, thus bringing errors into the data 2.) our focus was on a selected topic, while a vast majority of others was topic-independent. This meant that we had to take a different approach to removing the irrelevant data from the content of the selected topic.

We built a web platform capable of retrieving the public tweets using Twitter API v1.1 [4] based on a chosen hashtag. The tweet content was displayed to one of our evaluators who rated the tweet either as negative, neutral, positive or to be skipped if it was not an opinion. The evaluators were also asked to mark in the tweet content the crucial key words that determined the tweet rating, as during our data acquisition we were not sure whether our model would be capable of finding the meaningful and relevant words or phrases from the whole tweet content, given the small size of the training set. The tweet content was then saved into the database together with the information on the tweet ID, user ID, hashtag used for searching the tweets, key-words selected by the evaluator, the tweet rating, evaluator's name and time code. The basic steps of our data acquisition are shown in Fig. 1.

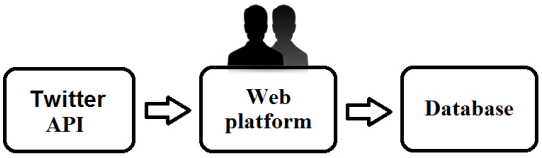

Figure 1. Basic data-acquisition steps.

The data for the training and testing dataset were acquired between 27th December 2013 and 4th January 2014 for the training and testing datasets and for another testing dataset four months later. The testing data were obtained so as to optimally simulate a real situation. This means that the assessment of some ten consecutive tweets of the selected hashtag was made over a period of approximately one hour. The N-Fold separation of the datasets was not preformed because of the complicated model-building process and the possibility of acquiring the data separately on our custom-built web platform. The hashtags on which our data-acquisition process was focused on were based on popular movies [13] viewed at the time and were as follows: #wolfofwallstreet, #3daystokill, #47ronin, #afterearth, #badgrandpa, #devilsdue, #dragonball, #endlesslove, #grudgematch, #ifrankenstein, #legomovie, #pompeii, #robocop, #springbreakers, #thehobbit, #totalrecall, #troll2 and #waltermitty. The total numbers of the gathered tweets divided by the sentiment for the three datasets are listed in Table 1.

	Training data	Test data #1	Test data #2
Negative	233	7	26
Neutral	111	6	11
Positive	758	78	62
All	1102	91	99

Table 1. Number of the tweets per rating/dataset.

The data were then exported from the server database to three different files; one for learning our model and two for testing it.

2.2 Data processing

To process the data and build a model we used Python 2.7.6 [15] with the NLTK 3.0. platform [14] (Natural Language Toolkit). Most of the text processing was done using the NLTK library which provided us with a useful word-tokenizer tool capable of detecting stop words and having a tool to extract the meaningful phrases from a given sentence. The tool enabling us to compare the results we got with our algorithm to common-approach method results was RapidMiner 5.3 with the Text Mining extension 5.3.2. The functions and classifiers described in Chapter 4.2 are already included either in RapidMiner or in the Text Mining extension.

3 MODEL

3.1 Raw-data preprocessing

Besides the meaningful text, the tweet content also contains several useless elements such as the URL addresses, special characters with no actual meaning and often misspelled words. In the basic preprocessing steps embedded into the web platform, we removed all URLs, special characters and all re-tweets (tweets re-posted by other users). We were able to detect the misspelled words, but we could not successfully correct them. We left them intact.

Before analyzing the data there was another preprocessing step necessary to be taken in addition to the one performed by the web-based platform. It was done in the subprogram generating the data files that were then exported from the server and used for further work. The special encoded characters used to work with the server database were replaced with the ones encoded in the UTF-8 format. Each emoticon used in the tweet content was replaced with words, each abbreviation was extended to its full length and the punctuation marks and special characters that had not been converted into words in the previous step were removed.

3.2 Model concept

Our main challenge was to build an independent model capable of working in a large variety of fields and being topic specific for a given training data. Researching the possibility of setting up a better and faster operating model, we developed a new approach unlike the ones described in Chapter 4.2. In our model the importance of the word phrases is greater than that of individual words. The basic steps taken in generating our model are shown in Fig. 2.

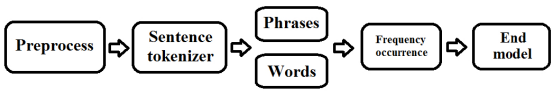

Figure 2. Basic steps in our model generation.

The preprocessing steps described in Chapter 3.1 were followed by splitting the content of each tweet into tokens using a general-purpose external tokenizer tool embedded in the NLTK library. The meaningful words were determined and the relevant phrases in the content were found and then saved. After obtaining the phrases and words from each tweet, the occurrence frequency was determined, properly weighted to give the phrases an appropriate relevance, and then used to build the end model.

The model was built as some kind of a dictionary. Every word and phrase maps the number indicating the power of presence in the tweet content of each class. The number is determined by examining the occurrence frequency of a word or phrase obtained on the basis of a statistical analysis of all the tweets. The larger the number the more often a particular feature occurs in the content. In common approaches this step is called the feature-extraction process, but after knowing the occurrence frequency, only a few features are taken to build the model, meaning that some information is taken no regard of. In contrast, our model works with all the features and the numbers of the corresponding occurrences without building it by using any standard model-building approach.

The model training is followed by setting up an internal database as in case of forming a dictionary. The tweet is classified by searching for words and phrases in the content from the generated dictionary and by converting words into a number. The feature numbers from the negative class are negatively signed, those from the positive class are positively signed and those from neutral class are not signed at all. A significant negative sum means that the tweet has a greater probability of being negative and vice versa. Anything close to zero is classified as the neutral class. The values for each class are then normalized to lie between -1 and 1, where -1 most certainly means negative, 0 a neutral and 1 positive rating of tweet. The observed classes can be represented with a sub-interval on the line as shown in Fig. 3.

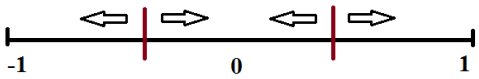

Figure 3. Optimizing borders between classes.

3.3 Model optimization

When building a model, there are three values that can be changed. Two of them are the numbers setting the borders between the classes as shown in Fig. 3, and one of them is the factor i.e., a numerical value determinining how many words are left out from the end model. By optimising the factor, it is possible to exclude topic-specific words, not-stop words, those present in every class and not uniquely characterizing an individual class. The factor number determines the size of the subinterval in the occurrence frequency distribution of each class. The higher the number the greater is the width of the interval in each class in which the same words are used. If, for example, a word appears in the first class seven times and in the second class only twice, the required value of the factor to exclude a word from the end model would be five. That means if in one class a word occurs multiple times and in the other one just a few times, it can or cannot be removed from the end model by varying the factor value.

After our model had been built with a certain factor, the borders were determined with an iterative process to find the best results. Arithmetic mean of the class recall value was chosen to provide the information about the model performance (1).

$$P[\%] = \frac{R_{positive} + R_{neutral} + R_{negative}}{3} \quad (1)$$

The above optimization criterion was chosen to ensure the model to perform correctly (that is, setting the borders and the factor) regardless of the biased data - the occurrence of the positive tweets is significant.

The results of the factor-dependant model performance are shown in Fig. 4.

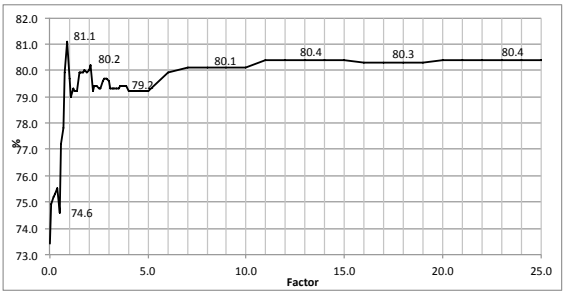

Figure 4. Factor-dependant success rate.

In the model-building process we experimented with four options of the word sets:

1) Using all the words from the tweet content
2) Using the meaningful words from the tweet content
3) Using all the words from the keywords marked by the evaluator
4) Using the meaningful words from the keywords marked by the evaluator.

The reasons for selecting this set of options are listed in Chapter 2.1. After performing the optimization on all four options and after testing the model, using all the words from the tweet content was proven as the best choice.

4 RESULTS

4.1 Final results

The results for the testing dataset #1 are listed in Table 3, and for the testing dataset #2 in Table 4. The results for the training dataset (self testing) are listed in Table 2.

4.2 Common-approach results

In order to compare the results we obtained by using our approach with those of the common approach, we decided to build a classification model with RapidMiner. We used the same data as when working with Python and similar text-processing algorithms (we transformed all the text to a lower case, tokenized it, filtered the stop words, stemmed the words, filtered the tokens by length and generated n-grams). This was all done in a block called Process Documents from Data. As this block is a part of the Text Mining extension in RapidMiner, it prepares a feature vector appropriate for machine learning with Naive Bayes and k-NN algorithms. After processing the text we applied a ten-fold cross validation with either the Naive Bayes or the k-NN classifier with the RapidMiner data mining tools. We then applied the trained model to our two testing datasets where RapidMiner then calculated the confusion matrices. However, the results obtained with the common approaches using the nearest neighbor and the Naive Bayes algorithms were quite different from the ones obtained with our approach. Confusion matrices for our training data are shown in Tables 5 (Naive Bayes) and 6 (k-NN), for our

Success rate: 84.21%				
	True negative	True neutral	True positive	Class precision
Prediction negative	207	11	20	86.98%
Prediction neutral	19	77	94	40.53%
Prediction positive	7	23	644	95.55%
Class recall	88.84%	69.37%	84.96%	

Table 2. Train-data (self-test) results.

Success rate: 61.54%				
	True negative	True neutral	True positive	Class precision
Prediction negative	4	2	5	36.36%
Prediction neutral	2	2	23	7.41%
Prediction positive	1	2	50	93.34%
Class recall	57.14%	33.33%	64.10%	

Table 3. Test data #1 results.

Success rate: 61.62%				
	True negative	True neutral	True positive	Class precision
Prediction negative	17	0	3	85.00%
Prediction neutral	18	6	21	17.14%
Prediction positive	1	5	38	86.26%
Class recall	65.39%	54.55%	61.29%	

Table 4. Test data #2 results.

Success rate: 74.14%				
	True negative	True neutral	True positive	Class precision
Prediction negative	146	36	48	63.48%
Prediction neutral	36	30	52	25.42%
Prediction positive	50	44	641	87.19%
Class recall	62.93%	27.27%	86.49%	

Table 5. Bayes training.

Success rate: 77.59%				
	True negative	True neutral	True positive	Class precision
Prediction negative	132	19	10	81.99%
Prediction neutral	27	26	33	30.23%
Prediction positive	73	65	697	83.47%
Class recall	56.90%	23.64%	94.19%	

Table 6. k-NN training.

first testing dataset in Tables 7 (Naive Bayes) and 8 (k-NN) and for out second testing dataset in Tables 9 (Naive Bayes) and 10 (k-NN).

4.3 Comparison between the results obtain with our approach and the common approach

A comparison between the success rate of our approach and the common approach can be seen on Fig. 5. Compared to our research, one might think that the Naive Bayes classifier performs better on the two testing datasets. However, it should be noted that all the Naive Bayes classifier did was to classify the tweets as positive and as there is a considerable class imbalance in the testing datasets, as seen from Table 1, it looks as if the success rate of this classifier were better than the one of our approach. The success rate was calculated with the formula given in equation (2).

Success rate: 67.03%				
	True negative	True neutral	True positive	Class precision
Prediction negative	0	0	1	0.00%
Prediction neutral	0	0	0	0.00%
Prediction positive	26	11	61	62.24%
Class recall	0.00%	0.00%	98.39%	

Table 7. Test #1 Bayes.

Success rate: 28.57%				
	True negative	True neutral	True positive	Class precision
Prediction negative	26	11	61	26.53%
Prediction neutral	0	0	1	0.00%
Prediction positive	0	0	0	0.00%
Class recall	100.00%	0.00%	0.00%	

Table 8. Test #1 k-NN.

Success rate: 72.73%				
	True negative	True neutral	True positive	Class precision
Prediction negative	0	0	1	0.00%
Prediction neutral	0	0	0	0.00%
Prediction positive	7	6	72	84.71%
Class recall	0.00%	0.00%	98.63%	

Table 9. Test #2 Bayes.

$$\text{Success rate}[\%] = \frac{\text{correctly classified tweets}}{\text{all tweets in dataset}} \cdot 100\% \quad (2)$$

However, taking into account the class imbalance calculated in equation (3) and calculating the success rate with the modified formula given in equation (4), we get a better perspective of the model efficiency (see Fig. 6). In equation (4), CRPC means the class recall per class (also the correctly classified tweets per class) and CIPC means the class imbalance per class. The class imbalance is calculated so that the rarer class has a bigger effect on the modified success rate.

$$\text{Class imbalance}[\%] = 1 - \frac{\text{number of tweets in class}}{\text{all tweets in dataset}} \quad (3)$$

$$\text{Success rate}[\%] = \sum \text{CRPC} \cdot \text{CIPC} \quad (4)$$

Success rate: 7.07%				
	True negative	True neutral	True positive	Class precision
Prediction negative	7	6	72	8.24%
Prediction neutral	0	0	1	0.00%
Prediction positive	0	0	0	0.00%
Class recall	100.00%	0.00%	0.00%	

Table 10. Test #2 k-NN.

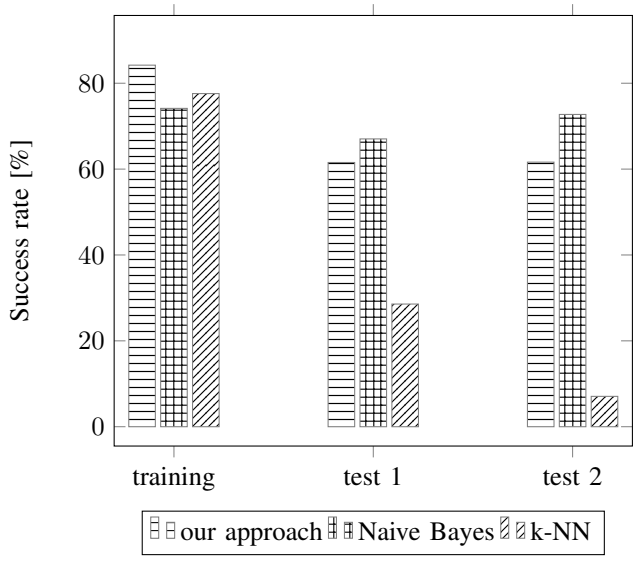

Figure 5. Comparison between methods on different datasets.

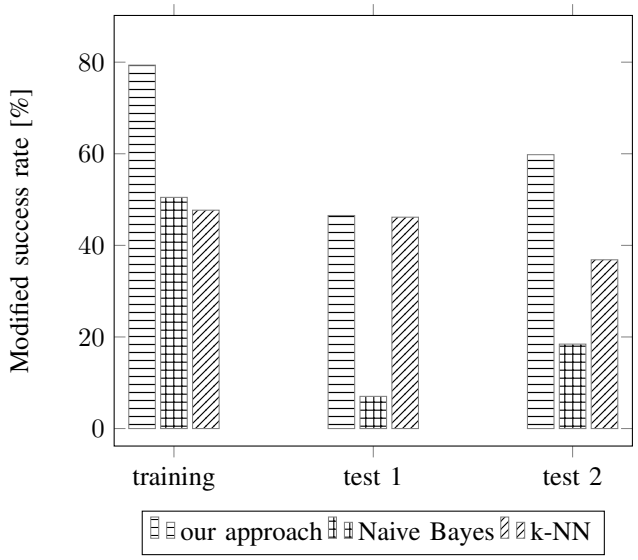

Figure 6. Comparison with modified success rate.

5 DISCUSSION

Our assumption that there is a third sentiment class, i.e. the neutral class was proven to be right. The tweets expressing the neutral sentiment are mostly the tweets praising some aspect of a movie but hating another. The defining word for such tweets has proven to be "but" (and all its synonyms).

However, we had a problem with a special sort of tweets, specifically the tweets about a movie expressing no sentiment but only an intent, e.g. *"I am going to see #wolfofwallstreet today"* , and the state of the person, e.g. *"Watching #legomovie with my sister."* Nevertheless, some of those tweets included links which we ignored (mostly because of the included promotional materials for a movie and no sentiment expressed). We believed that it would be wise to build a separate model to determine if a tweet does or does not include a sentiment.

With our data classification we proved that when analysing a text it is better to use the natural-language processing algorithms such as NLTK in Python. As seen from our results shown in Tables 2-10, the differences between the class recalls in our training datasets are not big, but there is a big difference in our two testing sets. Using our approach, the tweets can be classified into all three classes. This cannot be done by using RapidMiner with Naive Bayes and k-NN which can classify the tweets only into one of the three classes. We therefore believe that our approach using the natural-language processing tool is the right one, especially in short-text structures such as tweets where there are not that many options to express the same sentiment. With that in mind, we have to emphasize that this kind of an approach does not work for sarcasm.

The downside to this kind of text classification is the time it takes to gather a big dataset. Though not very big our dataset classifies most of the tweets in the correct class. If it were significantly larger, the results would be even better. However, when building the dataset, we encountered a great class imbalance. The tweets were mostly positive (as seen in Table 1). It can be concluded that people do not take time to tweet about something they do not like.

To end with, we believe this system can be useful as a reference to the public opinion tests, firstly for movies, and by expanding also to other topics which can be done by acquiring new training data. As mentioned in the introduction, to rate a movie (or any other topic), users have to use special pages, where they have to be registered. Using our approach means that the rating of a movie (or any other topic) could simply be measured by gathering the data already available on the world wide web without users having to do anything else that they have already done: tell their followers their opinion.

5.1 Future work

Of course, by improving it, our approach would perform even better. As said above the first improvement would be expanding the current training dataset. The next would be auto correcting the misspelt words. The same applies to using synonyms if words the model knows and understands as part of a class are not found inside the tweet content. An interesting area of the future work would also be profiling the users based on their opinions.

Comparing between the sentiment on Twitter and the sentiment on movie websites, such as IMDb or Rotten Tomatoes would also be one of the possibilities for improving the sentiment analysis. The limitation that the opinion on Twitter can be written with only 140

characters does not exist on either of those pages, thus enabling the texts to be longer and more descriptive.

Using WordNet-Affect offers another possibility for model improvement by gathering different words under the same emotion which is what we are also trying to do with the sentiment analysis. The next step would be obtaining a good text to speech machine, recording all the tweets in our datasets and using a speech sentiment analysis.

6 DISCLAIMER

The training and the two testing datasets used for this research were set up and manually classified by us. If you want to get access to the database, you are welcome to contact us. Both leading authors have equally contributed to the research.

REFERENCES

[1] Twitter, Inc. Twitter Help Center - Using hashtags on Twitter. https://support.twitter.com/entries/49309 (accessed May 22, 2014)

[2] Twitter, Inc. Discover Twitter - What is Twitter and how to use it. https://discover.twitter.com (accessed February 25, 2014).

[3] Twitter, Inc. About. https://about.twitter.com/company (accessed February 25, 2014).

[4] Twitter, Inc. Twitter developers - Documentation. https://dev.twitter.com/ (accessed May 2, 2014)

[5] Jansen, Bernard J., Mimi Zhang, Kate Sobel, Abdur Chowdury. Twitter Power:Tweets as Electronic Word of Mouth. *Journal of the American Society for Information Science and Technology* 62, Vol. 2 (2010): 406-418.

[6] Thelwall, Mike, Kevan Buckley, Georgios Paltoglou. Sentiment in Twitter Events. *Journal of the American Society for Information Science and Technology* 60, Vol. 11 (2009): 2169-2188.

[7] Neethu M S, Rajasree R. Sentiment Analysis in Twitter using Machine Learning Techniques. *Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference*: 4-6 July 2013: 1-5

[8] Go, Alec, Richa Bhayani, Lei Huang. Twitter Sentiment Classification using Distant Supervision. http://sentiment140.com/ (accessed January 5 2014).

[9] Ammar Hassan, Ahmed Abbasi, Daniel Zeng.Twitter Sentiment Analysis: A Bootstrap Ensemble Framework. *Social Computing (SocialCom), 2013 International Conference*: 8-14 September 2013: 357-364

[10] Ana C. E. S. Lima, Leandro N. de Castro. Automatic Sentiment Analysis of Twitter Messages. *Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference*: 21-23 November 2012: 52-57

[11] Seyed-Ali Bahrainian, Andreas Dengel. Sentiment Analysis and Summarization of Twitter Data. *Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference*: 3-5 December 2013: 227-234

[12] Miles Efron. Information Search and Retrieval in Microblogs. *Journal of the American Society for Information Science and Technology* 62, Vol. 6 (2011): 996-1008

[13] IMDb.com, Inc. Opening this week. http://www.imdb.com/ (accessed April 12, 2014).

[14] Natural language toolkit, NLTK Project. Rev. November 5, 2013. http://www.nltk.org (accessed May 2, 2014).

[15] Python software foundation. https://www.python.org/ (accessed May 2, 2014).

Aljaž Blatnik is a masters student at the Faculty of Electrical Engineering, University of Ljubljana. He is specializing in telecommunications. His areas of interest include social responses analysis and signal processing technologies.

Kaja Jarm is a masters student at the Faculty of Electrical Engineering, University of Ljubljana. She is specializing in telecommunications with a special interest in social data processing, user profiling and development of multimedia solutions.

Marko Meža has received his B.Sc. in 2001 and his Ph.D. in 2007 from the Faculty of Electrical Engineering, University of Ljubljana, both in electrical engineering. Currently he is working as a research scientist in LDOS, Faculty of Electrical Engineering, University of Ljubljana. Areas of his research include telemedicine, application of the machine learning algorithms to the specific problems in medicine and digital signal processing.