TakeLab at SemEval-2017 Task 4: Recent Deaths and the Power of Nostalgia in Sentiment Analysis in Twitter

David Lozić, Doria Šarić, Ivan Tokić, Zoran Medić, Jan Šnajder
Text Analysis and Knowledge Engineering Lab
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
{name.surname}@fer.hr

Abstract

This paper describes the system we submitted to SemEval-2017 Task 4 (Sentiment Analysis in Twitter), specifically subtasks A, B, and D. Our main focus was topic-based message polarity classification on a two-point scale (subtask B). The system we submitted uses a Support Vector Machine classifier with rich set of features, ranging from standard to more creative, task-specific features, including a series of rating-based features as well as features that account for sentimental reminiscence of past topics and deceased famous people. Our system ranked 14th out of 39 submissions in subtask A, 5th out of 24 submissions in subtask B, and 3rd out of 16 submissions in subtask D.

1 Introduction

Sentiment analysis (Pang et al., 2002), a task of determining polarity of text towards some topic, recently gained a lot of interest, mostly due to its applicability in various fields, such as public relations (Pang et al., 2008) and market analysis (He et al., 2013). Following the growing popularity of social networks and an increasing number of user comments that can be found there, sentiment analysis of texts on social networks, such as tweets from Twitter, has been the focus of much research.

However, determining the sentiment of a tweet is often not an easy task, since the length of the tweet is limited and language is mostly informal, including slang, abbreviations, and hashtags. Various systems have been proposed for tackling this problem, ranging from simple unsupervised models that use precompiled sentiment lexicons for evaluating polarity of tweets (O'Connor et al., 2010) to more complex supervised models that use textual feature representations in combination with machine learning algorithms such as Support Vector Machines (SVM) (Khan et al., 2015; Barbosa and Feng, 2010) or deep neural networks (Dos Santos and Gatti, 2014; Tang et al., 2014).

In this paper, we present our system for determining sentiment of tweets, which we submitted SemEval-2017 Task 4 (Rosenthal et al., 2017), more specifically to the English versions of subtasks A, B, and D. In subtask A, the goal was to predict the sentiment of a tweet as either positive, neutral, or negative. Subtask B consisted of predicting the sentiment of a given tweet on a 2-point scale (positive or negative) given a topic. In subtask D, the task was to determine the distribution of positive and negative tweets for each topic in a given set of tweets annotated with topics.

The system we submitted uses an SVM classifier with a linear kernel and a number of features. We experiment with basic features such as tf-idf and pretrained word embeddings, as well as more task-specific features including sentiment lexicons, ratings-based, "nostalgia features", and "recent deaths". Ratings-based features use external data from different online resources to leverage the information such as rating of a movie or an actor mentioned in a tweet. "Recent deaths" features make use of information about recent deaths of notable people, while "nostalgia feature" makes use of topic's "age" – the rationale being that people usually reminisce about past events in a sentimental and positive way. Our system ranked 3rd out of 16 teams in subtask D, 5th out of 24 teams in subtask B, and 14th out of 39 teams in subtask A.

2 Features

To build our model, we first preprocess tweets and extract various features. We use standard features such as bag-of-words (more precisely tf-idf), pre-

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 784–789,
Vancouver, Canada, August 3 - 4, 2017.

trained word embeddings, and count-based stylistic features. Additionally, we design task-specific features based on publicly available ratings for certain topics. We next describe the preprocessing and the features in more detail.

2.1 Preprocessing of Tweets

As a first preprocessing step, we tokenize tweets using a basic Twitter-adapted tokenizer.[1] After tokenization, we lemmatize and stem tweets and remove stopwords from each tweet using the NLTK toolkit (Bird et al., 2009).

Additionally, since some of our features require recognizing named entities in a tweet, we use the named entity tagger, also from the NLTK toolkit, for recognizing entities in tweet. Unfortunately, using NLTK-provided named entity tagger yielded unsatisfactory results, i.e., many of named entities in tweets were not recognized correctly. We assume this is due to tweets generally having poor capitalization, which is something named entity taggers in general are rather sensitive to.

As a remedy, we replaced the tagger with a greedy search algorithm. This approach simply looks for an occurrence of any string in tweet in some of our named-entity databases (introduced in the following sections). This proved to be a working solution for named entities longer than one word, but led to problems with unigrams, which were falsely recognized as named entities due to the existence of a movie or a game with that exact name. For instance, the word *"her"* would falsely be recognized as the 2013 movie *"Her"*.

Finally, we settled for a combination of the two approaches. We introduced parameters for the length range of word sequences. For the named entity tagger, we set the length range to $[0, 1]$, while for the greedy search algorithm we used the range $[2, 7]$. This way, we try to reduce the number of falsely recognized named entities in the greedy search algorithm (by omitting single word entities from its scope), while ensuring that some single word entities still get recognized by the named entity chunker.

2.2 Standard Features

We use a number of standard features typically used in sentiment analysis and other text classification tasks.

Word embeddings. For word embeddings we use GloVe (Pennington et al., 2014). We use 200-dimensional word embeddings, pretrained on 2B tweets. Final vector representation of a tweet is calculated as an average of the sum of vectors of all the words in a tweet.

Tf–idf. The standard tf–idf vectorizer from Python's scikit–learn package.[2]

Counting features. For each tweet we count the number of occurrences of various stylistic features: exclamation marks, question marks, elongated words, capitalized words, emoticons, and hashtags.

User information. We collected the information about authors of the tweets using the script provided by the organizers. From this data we extracted the number of followers, friends, and tweets of each user.

Sentiment polarity lexicons. We use sentiment lexicons developed by Bravo-Marquez et al. (2016), which contain three weights per word, indicating word's positive, neutral, and negative sentiment. These lexicons are built from automatically annotated tweets and existing hand-made opinion lexicons. We use the most positive word sentiment and the most negative word sentiment in each tweet as features, together with the number of extremely positive and extremely negative words in a tweet. A word is considered extremely positive if its positive weight in lexicon is higher than 0.75, and extremely negative if its negative weight is higher than 0.8.

2.3 Nostalgia Feature

We presume that some topics, such as games, movies, and music from years ago, are usually mentioned in positive light due to nostalgia.[3] To leverage this, for many of our topic-based features we try to take the age of the certain topic into account.

We use the following metric for calculating nostalgia feature, where applicable:

$$nost(y) = \min(m, y_{curr} - y) \qquad (1)$$

where y is the year of the content's release, y_{curr} current year, and m empirically determined upper bound for age.

[1] http://sentiment.christopherpotts.net/codedata/happyfuntokenizing.py

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

[3] Nostalgia is a sentimentality for the past, typically for a period or place with happy personal associations. (*Wikipedia*)

2.4 Ratings Features

We introduce a series of features that are calculated if a certain "rateable" topic is mentioned in a tweet. In order to build those features, we collected information from publicly available ratings for various domains: movies and TV shows, actors, games, musicians, historically influential people, and companies. It is worth noting that we are collecting these publicly available ratings independently of training the classifier, which makes our system applicable to tweets about new movies, TV shows, actors, etc.

Movies and TV shows. To gather movies' and TV shows' data, we used IMDb's publicly available plaintext database.[4] As the plaintext database is quite comprehensive, we filtered the data, leaving only movies and TV shows released in 2005 or later, with more than 50,000 user votes, and a minimum average rating of 4.0. This reduction left us with an acceptable amount of ~4,300 entries.

Movie-ratings features are implemented as a vector of 14 values: a binary value indicating if a movie was found in a tweet, movie's rating, number of user votes, movie's nostalgia value (as defined above), and 10 values representing user votes distribution per rating (from 0 to 9).

Actors. In the same manner as for the movies, we obtained the IMDb's plaintext database of actors using the same resource. We filtered out all actors that do not appear in the previously filtered movies database, to reduce the number of entries in an otherwise huge database. This left us with approximately 135,000 actor entries.

If an actor is mentioned in a tweet, his or her mention is represented with a single value in the feature vector – actor's rating. This rating is calculated by taking into account actor's appearances in various movies, as well as actor's position in movie's credits; the latter captures the intuition that each actor in a movie does not equally contribute to the movie's overall rating.

We calculated each actor's (a) rating for a single movie (m) as follows:

$$r(a,m) = r(m) \cdot \left(1 + c^{\frac{1-pos(a,m)}{k}-1}\right) \qquad (2)$$

where c is the percentage of the movie's rating taken into account, k is the rate of how much the position affects the rating, $r(m)$ is the movie's rating, and $pos(a,m)$ is the actor's position in the movie's credits. Actor's final rating is then defined as mean of their ratings in all the movies they participated in. During the evaluation, we set the hypeparameters c and k to 0.1 and 50, respectively.

Games. We obtained the data about video games by scraping *GameRankings*[5] website for all games with at least 10 reviews. This way we gathered about 8,500 game entries.

A mention of a game in a tweet is represented with three values in the feature vector: a binary value representing game's mention in a tweet, game's rating, and game's nostalgia metric, as defined in Section 2.3.

Musicians. For musicians' data, we scraped *Metacritic*'s *Music People* pages.[6] This gave us names, numbers of albums, and ratings for about 10,500 artists and bands.

Three values were added to the feature vector for musicians: a binary value representing musician's mention in a tweet, number of musician's albums, and musician's rating.

Important people. We use *MIT*'s *Pantheon*[7] list of historically influential people, with around 10,000 entries, to obtain a number of useful features. Based on this list, we compute 28 features, as follows:

- binary value indicating a person has been found in the obtained list;
- person's historic ranking on *Pantheon*;
- person's nostalgia value, derived from their birth year;
- person's Wikipedia page views;
- person's Wikipedia page views standard deviation;
- person's historic popularity;
- person's place of birth as a vector of 10 binary values representing the highest-ranked birth sites by *Pantheon*: U.S., U.K., France, Italy, Germany, Russia, Spain, Turkey, Poland, the Netherlands;

[4] ftp://ftp.fu-berlin.de/pub/misc/movies/database/

[5] http://www.gamerankings.com/

[6] http://www.metacritic.com/browse/albums/people

[7] http://pantheon.media.mit.edu/rankings/people/all/all/-4000/2010/H15

- person's occupation as a vector of 10 binary values representing the most historically influential occupations by *Pantheon*: Politician, Actor, Writer, Soccer Player, Religious Figure, Singer, Musician, Philosopher, Physicist, Composer.

Companies. Using *Good Company Ratings'* 2014 report,[8] we gathered the data about various companies, consisting of about 300 entries.

Company features contain four values: a binary value indicating company's mention in a tweet, its *Fortune* rank, seller ratings, and steward ratings.

2.5 Recent Deaths

Due to the impact celebrity deaths have on social media and the fact that people usually reminisce in a positive way about deceased people, we posit that the information whether a person mentioned in a tweet died recently would prove useful for sentiment analysis. To this end, we gathered from *Wikipedia*[9] a list of significant people who died in the last three years.

We represent deaths using a single value indicating the number of years that have passed from the person's death. While at first similar to nostalgia metric introduced above, the death metric accounts for recent events and, therefore, emphasizes values that are the opposite of the values obtained with nostalgia metric. The death metric is given by $1 - nost(y)$, where y is the year of death, with m set to 3 (last three years only).

2.6 Controversy

We encode controversial topics as 41 binary values, one for each of the currently controversial events listed in the *University of Michigan-Flint*'s *Frances Willson Thompson Library*.[10] To further improve the performance of correct identification of controversial topics, we additionally provide alternative phrases for some of the issues, for example *"Affordable Care Act"* is also triggered by its popular nickname *"Obamacare"*.

2.7 Curse Words

We use a list of 165 curse words often used in tweets, compiled by Kukovačec et al. (2017). Presence of a curse word in a tweet is encoded with a single binary value, indicating whether a curse word was found in tweet or not.

2.8 Topics and Hashtags

We built the above-mentioned features for three cases: topic identified from tweet's text, topic explicitly given as tweet's topic in subtasks B and D, and for the topic that might appear in a hashtag as a part of tweet.

Since hashtags usually contain more than one word in a single string, we adapt the greedy splitting procedure proposed by Tutek et al. (2016), which uses a dictionary of known words to split a hashtag that is a concatenation of multiple words. Additionally, for each word obtained from hashtag splitting we generate sentiment lexicon features, acknowledging that sentiment is often expressed via a hashtag.

3 Evaluation

We started with a number of different classifiers and chose the one that gave the best result on a hold-out test set (2016 test set) (Nakov et al., 2016) for each subtask. After the official results were published, together with the gold labels for test sets, we additionally performed a simple feature analysis over predefined feature groups to analyze the impact each of these groups has on the final result.

3.1 Evaluation Metrics

We submitted our solution to three subtasks: A, B, and D. Subtask A uses macro-averaged recall (ρ) over all three classes (positive, neutral, and negative) as an official metric. For subtask B, macro-averaged recall over positive and negative classes (ρ^{PN}) is used, while subtask D uses Kullback-Leibler Divergence (KLD) as the official measure.

3.2 Classifier Selection

We experimented with a number of classification algorithms from the scikit-learn package (Pedregosa et al., 2011): Support Vector Machine with a linear kernel (SVM), Logistic Regression (LR), Multinomial Naive Bayes (MB), Random Forest (RF), and a Stohastic Gradient Descent classifier (SGD). For training, we used all the available data provided by the organizers, except for the 2016 test set, which we used for testing the classifiers. We performed 5-fold cross-validation

[8] http://www.goodcompanyindex.com/ratings/

[9] http://en.wikipedia.org/wiki/Lists_of_deaths_by_year

[10] http://libguides.umflint.edu/topics/current

Classifier	Subtask A (ρ)	Subtask B (ρ^{PN})
LR	0.620	0.742
NB	0.467	0.663
RF	0.487	0.549
SGD	0.488	0.614
SVM	**0.623**	**0.752**

Table 1: Results for subtasks A and B on 2016 test set using the official measures for both subtasks

on our training set for optimizing each classifier's parameters. We used all of the described features for evaluating classifiers for subtask B and a subset of features for subtask A (everything except rating features extracted from explicit topic, features extracted from hashtags, user information features, curse words presence, and controversial topics).[11]

We observed that macro-averaged recall in subtask B (the official measure) improved (from 0.705 to 0.752) when we completely excluded the ratings and user information features, although the accuracy notably dropped (from 0.899 to 0.855). For this reason, we decided to submit our solution to both subtasks B and D without those two feature groups, since it led to higher recall at the expense of lower overall accuracy.

Table 1 shows the results in terms of macro-averaged recall on the 2016 test set for subtasks A and B for all of the models we experimented with and the final set of features that we included in the final submissions. We chose SVM with a linear kernel for all of our submissions, as it gave the best result on both subtask A and B. For subtask D, we used the outputs of the classifier built for subtask B and calculated the distribution of tweets using a simple "classify and count" approach.

Our submissions ranked 14th on the leaderboard for subtask A, 5th for subtask B, and 3rd for subtask D. Table 2 shows scores of top 7 submissions for subtasks B and D.

3.3 Feature Analysis

After the testing phase finished, we carried out an analysis of the impact of the specific feature groups in classification, for each of the three subtasks we took part in. We performed an ablation study over all feature groups. The results of these analyses are shown in Table 3.

[11] Unfortunately, we did not finish our system in time for the submission of subtask A, which resulted in differences between submissions for subtask A and the other two subtasks.

Team	ρ^{PN}
BB twtr	0.882
DataStories	0.856
Tweester	0.854
TopicThunder	0.846
TakeLab	**0.845**
funSentiment	0.834
YNU-HPCC	0.834

Team	KLD
BB twtr	0.036
DataStories	0.048
TakeLab	**0.050**
CrystalNest	0.056
Tweester	0.057
funSentiment	0.060
NileTMRG	0.077

Table 2: Official results for subtasks B and D (top 7 teams only)

Excluded group	A (ρ)	B (ρ^{PN})	D (KLD)
None	0.615	0.849	0.054
Counting	0.616	**0.852**	**0.050**
Lexicon	**0.617**	0.850	0.052
Ratings	**0.617**	0.846	0.053
Tf-idf	0.610	0.840	0.061
User info	0.615	0.849	0.053
Word2vec	0.602	0.798	0.116

Table 3: Feature analysis results for all subtasks

The analysis confirmed that excluding some feature groups indeed helps in obtaining higher recall in both subtask A and B. More specifically, excluding counting or lexicon features improved recall in subtask B, while excluding counting, lexicon, or ratings features from the set of features used for subtask A led to an increase in recall as well. Moreover, KLD score improves as well, when the group of counting features is excluded from complete set of features.

4 Conclusion

In this paper we described our solutions to SemEval 2017 Task 4 – subtasks A, B, and D. Our solution is based on a linear SVM classifier with some standard and a series of task-specific features, including rating-based features obtained from various websites as well as features that account for sentimental reminiscence of past topics and deceased famous people. Our system performed relatively well in all three subtasks. We ranked 14th out of 39 in subtask A, 5th out of 24th in subtask B, and 3rd out of 14 in subtask D.

For future work, it would be interesting to expand our model's feature set to non-covered domains (e.g., sports), and also to investigate how our model behaves on a more diverse set of topics. Expanding the system with a topic classifier as a pre-sentiment processing step might also be worth investigating, since the way sentiment is expressed varies across different domains.

References

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on Twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 36–44.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Felipe Bravo-Marquez, Eibe Frank, and Bernhard Pfahringer. 2016. Building a Twitter opinion lexicon from automatically-annotated tweets. *Know.-Based Syst.* 108(C):65–78.

Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*. pages 69–78.

Wu He, Shenghua Zha, and Ling Li. 2013. Social media competitive analysis and text mining: A case study in the pizza industry. *International Journal of Information Management* 33(3):464–472.

Aamera ZH Khan, Mohammad Atique, and VM Thakare. 2015. Combining lexicon-based and learning-based methods for Twitter sentiment analysis. *International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE)* page 89.

Marin Kukovačec, Juraj Malenica, Ivan Mršić, Antonio Šajatović, Domagoj Alagić, and Jan Šnajder. 2017. Takelab at semeval-2017 task 6: #rankinghumorin4pages. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 395–399. http://www.aclweb.org/anthology/S17-2066.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 Task 4: Sentiment analysis in Twitter. *Proceedings of SemEval* pages 1–18.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. *ICWSM* 11(122-129):1–2.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, pages 79–86.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2(1–2):1–135.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 501–516. http://www.aclweb.org/anthology/S17-2088.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A deep learning system for Twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. pages 208–212.

Martin Tutek, Ivan Sekulić, Paula Gombar, Ivan Paljak, Filip Čulinović, Filip Boltužić, Mladen Karan, Domagoj Alagić, and Jan Šnajder. 2016. TakeLab at SemEval-2016 Task 6: Stance classification in Tweets using a genetic algorithm based ensemble. *Proceedings of SemEval* pages 464–468.