A Sentiment Analyzer for Informal Text in Social Media

Huong Thanh Le, Nhan Trong Tran
Hanoi University of Science and Technology, Vietnam
huonglt@soict.hust.edu.vn, ttnhan92@gmail.com

ABSTRACT

This paper introduces our approach to Twitter sentiment analysis, with the task of classifying tweets as positive, negative or neutral. In the preprocessing task, we propose methods to deal with two problems: (i) repeated characters in informal expression of words; and (ii) the affect of contrast word in determining sentence polarity. We propose features used in this task and investigate an optimal method of using these features. Classification algorithms including Decision Tree, K Nearest Neighbor, Support Vector Machine, and a Voting Classifier are used for implementing the system. Experiment results with Twitter 2016 test dataset shown that our system achieved good results (63.7% F1-score) compared to related research in this field.

CCS CONCEPTS

I.2 [Artificial Intelligence]: Natural Language Processing

KEYWORDS

sentiment analysis, word embedding, decision tree, kNN, SVM, Voting Classifier

1 INTRODUCTION

Recently, social networking sites such as Facebook and Twitter become more and more popular with millions of users sharing either information or opinions about personalities, politicians, products, and events every day. They are valuable resources for business analysis, marketing, social analysis, etc. Companies use such information for improving their products or adjust marketing strategies. Government and politicians interest in citizens' opinions about current political and social events. Because of that, Twitter sentiment analysis has received a lot of interest from research community.

The task of sentiment analysis is to classify a review into one from some predefined categories. Early works in sentiment analysis deals with long text such as product review, movie review, restaurant reviews etc. The system has to determine whether such an expression is positive, negative, or neutral. Classification algorithms such as Support Vector Machines (SVMs) [1] work well with sentiment analysis at this level since each document is well-written and long enough for representing as a bag-of-words.

Exploring the sentiment of tweets is more challenge than those typically encountered when working with traditional text. The main difficulties are:

1. Tweets are short. The size of a tweet is limited to 140 characters, which provides not enough information for classification algorithm working correctly.
2. The language used is very informal, with creative spelling and punctuation, misspellings, slang, new words, URLs, genre-specific terminology, abbreviations and #hashtags. Such informal words make tweets ambiguous and difficult to understand. For example, "4" can be understood as the number "four" or the preposition "for".

Examples below illustrate a part of these difficulties:

Example 1: *Haha... I want to see. E macdonalds here cheaper. Yum yum.*

Example 2: *Ya... She wans... But now so late dunno still can arrange 4 tmr anot...*

The sentiment of Example 1 can be recognized as positive basing on words "want", "cheaper", "yum yum". Example 2 is harder to automatically analyze since it contains many informal words, "ya", "wans", "dunno", "4", "tmr", "anot", which are intepreted as "yes", "wants", "don't know", "for", "tomorrow", "or not", respectively. This example is considered as negative basing on words "late" and "dunno".

The difficulties mentioned above have reduced the system performance dramatically when applying traditional approaches in sentiment analysis. In this paper, we present our approach to Twitter sentiment analysis, with the task of classifying tweets as positive, negative or neutral. We concentrate on reduce the **effective** of the above mentioned problems in the preprocessing task. We also investigate a method to deal with contrast words in determining sentence polarity. We propose features used in sentiment analysis and investigate an optimal method for implementing these features in order to obtain the best outcome. Classification algorithms including Decision Tree (DT), K Nearest Neighbor (kNN), Support Vector Machine (SVM) are chosen as classification algorithms of the system. Since a tweet can be classified differently by different algorithms, a voting algorithm is used to vote from the above mentioned classifiers, in order to get more reliable results.

The remainder of this paper is organized as follows. Section 2 introduces related works to sentiment analysis. Word embeddings, a technique that is used in our system, is briefly

described in Section 3. Section 4 discusses about our preprocessing step, proposed features and classification algorithms. Section 5 describes our experimental results with different strategies to combine features. Section 6 concludes the paper and proposes directions for future work.

2 RELATED WORKS

The majority of research in sentiment analysis use machine learning approaches with different feature types. Those features include ngrams, part of speech, sentiment words, rules, word position, length measures, adjectives and adverbs, interjections and question marks, syntactic structures, semantic relations, etc. Grammatical features such as adjectives and adverbs, syntactic structures, semantic relations are potential indicators for sentiment analysis. Adjectives and adverbs carry most of the information regarding sentiment of a document. Hence, using these as features can produce good classification results [9]. However, they can only be applied in formal languages like newswire text. Tweets with many special characters, informal words, and ungrammatical structures cannot take advantages of such features.

Several efforts have been made to solve this problem. Kiritchenko et al. [6] developed a linear-kernel Support Vector Machine classification using a variety of surface form, semantic, sentiment, and negation features. The sentiment features were primarily derived from novel high-coverage tweet-specific sentiment lexicons. These lexicons were automatically generated from tweets with sentiment-word hashtags and from tweets with emoticons.

Deshwal and Sharma [5] combined several feature types like emoticons, exclamation and question mark symbol, word gazetteer, unigrams and testing on six supervised classification algorithms.

SemEval (Semantic Evaluation) - an ongoing series of international workshop on semantic evaluation - has been organized since 2007 with different tasks. Sentiment Analysis in Twitter has become one of SemEval's tasks since 2013 with more than 30 systems participated each year. One of its tasks is to determine whether a tweet is positive, negative or neutral. The task provides tweet datasets for training and testing classification systems. Some typical systems in SemEval2016 are introduced in the remaining part of this section.

As reported by Nakov et al. [10], the top four systems in SemEval 2016 extends the Convolutional Neural Networks (CNN) and seven out of top ten systems used either general purpose or task-specific word embeddings, generated via word2vec or GloVe.

Rouvier and Favre [7] introduced learning polarity classifiers for three types of embeddings, based on the same CNN architecture. Each set of word embedding modelled the tweet according to a different point of view: lexical, part-of-speech and sentiment. A final fusion step was applied, based on concatenating the hidden layers of the CNNs and training a deep neural network for the fusion.

Xu et al. [11] used a soft voting ensemble of a word2vec language model adapted to classification, a convolutional neural network (CNN), and a long short term memory network (LSTM).

Aueb [12] - the system standing at the fifth position at the campaign - used supervised learning with GloVe word embeddings for Twitter and weighted ensemble of classifiers.

Hamdan [13] proposed a sentiment analysis system using Logistic Regression classifier with a wide range of features including unigram, bigram, trigram, negation, sentiment lexicons, and semantic features (Brown clustering). All terms with occurrence less than three were removed from the feature space. Negation features were used to handle the negated context. Sentiment lexicons features contained information about positive and negative words in a tweet. Semantic features were used to reduce the sparsity of context representation.

Lango et al. [14] used Random Forests, SVMs, and Gradient Boosting Trees for the classification task, with a feature set including ngrams, Brown clustering, sentiment lexicons, WorldNet, and part-of-speech tagging. NLTK WordNetLemmatizer was used in the preprocessing step to get the stemmed form of words.

In this paper, we propose a sentiment analysis system using several supervised learning algorithms. Different features and their combinations are investigated to choose the optimal feature set for the sentiment analysis task. The system relies on word embeddings to deal with informal expression and to compute semantic meaning of words. Word embeddings will be introduced next.

3 WORD EMBEDDINGS

Word embedding is a technique to map words or phrases from a vocabulary to a vector of real numbers. This representation is more efficient and expressive than the traditional bag-of-words. The bag-of-words approach, especially in the case of representing tweets, often results in huge, very sparse vectors, where the size of each vector is equal to the vocabulary size. Word embedding aims to create a vector representation with a much lower dimensional space. Basing on the idea that words appearing in the same contexts share the same meaning, words are embedded in a vector space where semantically similar words are located to nearby points.

FastText [15] is a commonly used model for word embedding. It is an extension of word2vec, created by Facebook. It uses a fast and effective method to learn word representations and perform text classification. It has released pre-trained word vectors for 294 languages, trained on Wikipedia. However, these word vectors are not good for our task since Wikipedia and Twitter use different text types. Because of that, we create our own model in 300 dimensions by training FastText on Sentiment140 [1] [16] - a large Twitter dataset with many word extensions created by repeating some of its characters (e.g.,

[1] Available at http://help.sentiment140.com/for-students

"hello" vs. "hellllooooo"). This dataset is preprocessed by replacing all three or more duplicate consecutive characters with two (e.g., niccccceeee to niccee) as described in Section 4.1 before being trained. The purpose is to reduce the vocabulary of Sentiment140 before training, in order to have a more concrete representation of word vectors.

4 PROPOSED APPROACH

4.1 Preprocessing

As mentioned in Section 2, understanding tweets is challenge since many informal expressions with numerous spelling errors, url and emoticon are used. Therefore, a crucial task is to preprocess tweets to reduce text's ambiguities. It helps to reduce the tweets' representation space and to increase the similarity between two similar tweets written in two different ways. Our preprocessing task includes of the following steps:

- Step 1: Lowercasing all the input text;
- Step 2: Converting all url to ÜRL and @username to AT_USER;
- Step 3: Converting all abbreviations, slang and emoticons to their meaning (e.g., :) to “happy”, “dunno” to “don’t know”);
- Step 4: Removing all duplicate whitespace;
- Step 5: Replacing all three or more duplicate consecutive characters with two (e.g., niccccceeee to niccee).
- Step 6: Extracting the main clause in a tweet having a contrast relation

Since steps 1, 2, and 4 are simple, only step 3, 5, and 6 are described in the rest of this section.

Step 3: Converting all abbreviations, slang and emoticons to their meaning

To get the meaning of abbreviations, slang and emoticons, a Twitter dictionary is manually constructed from Webopedia Twitter dictionary [2] (including 119 Twitter slang words and abbreviations) and other twitter corpora. A part of our Twitter dictionary is shown in Table 1 below.

Table 1: A part of Twitter Dictionary

Twitter expression	Meaning
:)	Happy
:(	sad
wat	what
hee	here
u	you
r	are

Abbreviations, slang and emoticons can be solved partly by using a Twitter dictionary. However, the Twitter dictionary is never completed since new abbreviations are created everyday and there is no rule to generate such slang and abbreviations. Another solution to this problem is to learn word meaning from a large training data. Words need to appear frequent enough to be learned by the system. Beside the Twitter dictionary, word2vec model is also used in our system to get the actual meaning of slang and abbreviations.

[2]http://www.webopedia.com/quick_ref/Twitter_Dictionary_Guide.asp

Step 5: Replacing all three or more duplicate consecutive characters with two

Another case of informal words is word extensions being created by repeating some of its characters (e.g., hellllooooo). Several solutions have been used by previous reseearch to solve this problem. The simplest way is to use predefined rules to normalize misspelling words by convert all repeat characters into one. For example, 'yeeesss' is changed to 'yes'. However, this approach also change correct word into incorrect one (e.g., 'too' vs. 'to', 'loop' vs. 'lop', ‘hello’ vs. ‘helo’, etc.). We call this situation as over-normalization.

Hamdan [13] addressed this problem by using Brown corpus. This corpus contains 1000 hierarchical clusters over 217 thousand words. Original words and theirs extensions are kept in one cluster (e.g. yes, yess, yesss, yep). However, the Brown corpus cannot foresee and store all words' extensions (e.g., yeeeesssssss). As a result, these words are unrecognized by the system.

Rouvier and Favre [7] and Xu et al. [11] solved the problem of informal expressions by using word embedding, based on the idea that words appearing in the same context must share the semantic meaning. However, many variants of words still cause the sparseness of the feature space, thus reduce the system's learning capability.

To solve the above mentioned problems (unforeseeable/ new words and over-normalization), first we remove all repeat characters in a word until two repeat characters are remained. The output of this step still contains misspelling words, which are not in a word dictionary. However, this method can reduce the representation space of tweets. Word vectors generated by Fasttext word2vec are then applied to get the semantic representation of words. At this point, words with similar meaning and theirs extensions will be located nearby in the semantic space.

Step 6: Extracting the main clause in a tweet having a contrast relation

In natural language, contrast relation is used to connect two or more clauses with contrast meaning.

For example:

I thought it was good, *but* it was awful.

The first clause of the about sentence is positive, however the sentence is negative as the second clause is negative.

Since tweets often are ungrammatical sentences, we do not sepatate clauses in a tweet based on a syntactic parser. Instead, contrast words such as "but", "however", "on the contrary", … are used to do this task. If there is a contrast word in a sentence, the text after this word determines the sentiment polarity of the

sentence. Therefore in this step, if a sentence contains a contrast word, the sentence is replaced by the text after that word. A list of contrast words is manually created in our system.

4.2 Feature Selection

Different features have been implemented and tested in our system in order to choose the most useful features for sentiment classification. Our proposed features are introduced next.

4.2.1 Word unigrams. Bag-of-Words is one of the most successful feature representations in text categorization tasks. It is also used in sentiment analysis (e.g., [13,14]) to classify sentiment polarity, with each tweet being represented as a vector of unigrams.

This feature is also used in our system to test the effectiveness of unigram in sentiment classification.

4.2.2 Semantic features. Since tweets are very short and containing various modifications of words, representing tweets as vectors of unigrams as in some previous research (e.g., [13,14]) will give us a large and spare vector space, which will slow down the classification process and result in inaccurate predict.

To solve this problem, instead of representing each tweet by a bag of unigrams, semantic meanings of these words are used.

Based on our word2vec model trained by Fasttext mentioned in Section 3, semantic values of all words in a tweet are summed by each dimension to get values for semantic features of the tweet. All tweets are now represented by 300 dimension-vector containing information about semantic meaning of the tweet.

4.2.3 Sentiment feature. The sentiment score of a tweet is calculated by summing word-sentiment associations of this tweet. SentiWordNet [17] are used to get word-sentiment. SentiWordNet is a lexical resource for sentiment analysis which assigns to each synset of WordNet three sentiment scores - positivity, negativity, objectivity - between 0.0 and 1.0. It is used to find semantically related words and to get words' sentiment scores. Sample entries of SentiWordNet can be found in Table 2.

Table 2: Sample SentiWordNet Entries

POS	ID	PosScore	NegScore	SynsetTerms	Gloss
a	01740	0.125	0	able#1	(usually followed by `to') having the necessary means …
a	19731	0.125	0.125	handy#1	easy to reach ...

In the above table, each line contains information about part-of-speech, synset's ID, positive score, negative score, synset term, and glossary. POS with the value 'a' means that the synset is an adjective.

The sum of positive scores and the sum of negative scores are added to the feature vector.

4.2.4 Negation feature. Negation words such as “not”, "cant", and "never" can change the sentiment of a sentence from positive to negative and vice versa. Therefore, this is an important feature in sentiment classification.

Some research uses question mark ("?") as a negation feature. However, our empirical study find that it is not always the case. For example, the statements "*Why am I feeling worse*", "*y everything is so hard*" are negative statements; "*Why am I feeling worse?*" and "*y everything is so hard?*" are still negative notions. Therefore, question mark is not used as a feature in our classification system.

We has implemented the negation feature by two ways: one as a post processing rule and another as a feature in the feature set.

In the former case, after the polarity of input tweet has been predicted by the classifier, the post processing rule is applied to reverse the sentiment polarity of the tweet. In the later case, if a sentence contains negation words, the negation feature is 1, and 0 if otherwise.

To detect negation words, a negation dictionary is manually constructed from Sentiment140 dataset, including 19 negation words and symbols.

4.3 Classification Algorithms

We consider the task of classifying a tweet as positive, negative, and neutral. Four classifications algorithms have been chosen, including decision tree, k Nearest Neighbor, Support Vector Machines, and a Voting Classifier.

Decision Tree. Decision tree learning is a method to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

k Nearest Neighbors. The k Nearest Neighbors model looks at the k closest neighbors to the given sample and uses a voting mechanism to classify the given sample into the class with the majority of votes. By empirical study different values of k, the number of neighbors (k) is set to 24, which gave us most accurate results.

Support Vector Machine. Support Vector Machine has proved to be one of the most effective classification algorithms for the task of sentiment analysis. The idea of SVM is to find a hyper plane that has the largest distance to the nearest training-data point of any class, since the larger the margin the lower the generalization error of the classifier.

The original version of SVM separates data into two classes. In classifying tweets into three classes, we utilize a multiclass SVM in scikit-learn[3] and libLinear[4].

[3] A machine learning library for data mining and data analysis, available in the website http://scikit-learn.org

Voting Classifier. This classifier is a type of "Ensemble Learning" where multiple learners are employed to build a stronger learning algorithm. Beside the above three classification algorithms, we use a Voting Classifier as a soft voting method to predict the class labels by averaging the class-probabilities which taken from the outputs of decision tree, kNN, and SVM. The soft voting for each tweet is computed as:

$$y_{\text{Voting Classifier}} = \text{argmax}_v(\sum_i w_i * p_{i,v}) \quad (1)$$

where w_i is the weight of the classifier i; $p_{i,v}$ is the probability that the classifier i assigning sentiment polarity v for the input tweet. $w_i \geq 0$ and $\sum_v p_{i,v} = 1$ for $\forall$i.

5 EXPERIMENTS

5.1 Dataset

Three Twitters datasets were used in our experiments: Sentiment140, Twitter 2013 in SemEval2013 and Twitter 2016 in SemEval2016 for task 4, subtask A[5]. Sentiment140 dataset with 1.6 millions tweets was used to train by word2vec model to get its word embedding.

Twitter 2013 and Twitter 2016 training and developing dataset were used to train our sentiment classifiers. The total data in two Twitter training datasets is more than 15000 samples. Each sample has a link for retrieving data from Twitter. However, some of the links were no longer available on Twitter. As a result, only 19337 tweets are retrieved with 8152 positives, 8133 neutral, and 3052 negatives.

For the test dataset, 3547 tweets are retrieved from 3813 ones in Twitter 2013 test dataset; 20632 tweets were retrieved form Twitter 2016 test dataset with no tweet unavailable.

Since the size of Twitter 2013 test corpus we can get is smaller than actual dataset used in SemEval 2013 competition, we cannot directly comparable our result with other research used Twitter 2013 test dataset. Therefore, only Twitter 2016 dataset were used for evaluating our system performance.

The detail description of the data available for download is given in Table 3.

Table 3: Statistics of the Successfully Downloaded Part of the SemEval 2013 and SemEval 2016 Twitter Sentiment Classification Dataset.

Dataset	Total	Posit.	Negat.	Neutr.
Twitter 2013 (train)	9684	3640	1458	4586
Twitter 2013 (dev)	1654	575	340	739
Twitter 2016 (train)	6000	3094	863	2043
Twitter 2016 (dev)	1999	843	391	765
Our training data	**19337**	**8152**	**3052**	**8133**
Twitter 2016 (test)	**20632**	**7059**	**3231**	**10342**

[4] A library for Large Linear Classification, available at the website https://www.csie.ntu.edu.tw/~cjlin/liblinear/.

[5] Since we are unable to get Twitter dataset in SemEval 2017, the datasets in SemEval 2013 and SemEval 2016 are used in our experiments.

5.2 Experimental Setting

Experimental results were evaluated using macro-averaged F1-score. The first experiment was carried out to find the best algorithm among four classification algorithms mentioned in Section 4.3. Our proposed feature sets used in this experiment including semantic features, sentiment features, and negation feature. Table 4 presents our system performance with four classifiers (i.e., decision tree, kNN, SVM, and a Voting Classifier), which taken from scikit-learn library.

Table 4: Our System Performance with Four Classifiers

	F1-score (%)
DecisionTree	52.2
KNN	57.0
SVM	59.6
Voting Classifier	63.7

Table 4 points out that SVM is the best among three classifiers decision tree, kNN, and SVM.

The weight w_i of each classifier (i.e., decision tree, kNN, SVM) were optimized during the training time of Voting algorithm. Different sets of weights have been tested using the training data. The best values are $w_{DT} = 1$, $w_{kNN} = 1$, $w_{SVM} = 2$. Experimental results shown that the Voting Classifier provided a better result than SVM with the F1-score 4.1% higher.

Beside two main difficulties in analyzing tweets as mentioned in Section 1, another reason for the low F1-score of sentiment analyzing systems in general is that tweets (and maybe other text types) often contain a mix of positive and negative sentiment. For example, the text "*Yup no more already... Thanx 4 printing n handing it up.*" can be classified as either positive or negative sentiment. Putting such a tweet in only one class (e.g., positive, negative) will reduce the system accuracy.

To test the effective of using unigrams or their semantic replacements as features in sentiment classification, we carried out two experiments with our best classifier - Voting Classifier. One experiment used semantic features, sentiment features, and negation feature. Another experiment replaced semantic features by unigrams. One problem with our second experiment was that using unigrams resulted in a spare feature space. Since SVM in scikit-learn did not deal well with such problem, libLinear library was used instead.

Experimental results are shown in Table 5 below.

Table 5: Our System Performance with Different Feature Sets

	F1-score (%)
Sem+Sen+Neg	63.7
unigram+Sen+Neg	55.2

Table 5 proves that using semantic features instead of unigrams does not only reduce the representation space but also improve the system performance (from 55.2% to 63.7%). It confirms that replacing unigrams by semantic features is a good choice in the sentiment analysis task for social network text.

In order to choose the best way of using the negation feature, another experiment was carried out. In this experiment, negation was not used a feature in the feature set. Instead, it was implemented as a hand-written rule in the post-processing step:

Negation rule: If the sentence contains a negation word, its sentiment polarity will be reversed.

The F1-score in this case using our Voting Classifier was 61.1%, lower than the case using negation as a feature (63.7%).

To investigate the effect of contrast word, we removed step 6 from the preprocessing task, retrained and tested the system with this new preprocessing module using the Voting Classifier. The F1-score in this case is 63.5%, reducing a little bit comparing to the case using contrast words (63.7%). It indicates that using contrast words has a positive effect in preprocessing step.

Analyzing system outputs has shown that the text before the contrast word can be used to determine the sentence polarity when the sentiment polarity of the text after the contrast word is unclear. We believe that integrating this idea into our system can promote the system performance further. This will be one of our future works.

Our experiments with different implementations gave us the best result of 63.7%, when using our Voting Classifier with the feature sets: semantic features, sentiment features, and negation feature.

Comparison with other systems

Results of SemEval2016 competition prove that deep learning is the most powerful approach, with all top four systems use deep neuron networks. Note that each research used a different training set. Sensei-LIF [7] used the train and development corpora from Twitter 2013 to 2016 for training and Twitter 2016-dev as a development set. Beside Twitter dataset from SemEval2016 subtask A, Unimelb [11] also used training data from SemEval 2016 subtask 10. To pretrain their network using distant learning, they took a random sample of 10M English tweets from a 5.3TB Twitter dataset crawled from 18 June to 4 Dec, 2014 using the Twitter Trending API. Aueb [12] used 19,305 tweets from the 2016 datasets SemEval-2016 Task 4, as well as data from SemEval-2013 Task 2. Therefore, we did not seek for systems using the same training set like us. Instead, our system and the systems that we compared with must have the same test set (Twitter 2016).

Since our research concentrates on improving preprocessing task, investigating and proposing important features for classification algorithms, deep learning is not used in our system. However, Table 6 shows that our F1-score is still higher than the 1st rank in SemEval 2016 campaign. It proves that our proposed preprocessing module, the quality of our features, as well as our proposed soft-voting method in the Voting Classifier are potential candidates for improving performance of a Twitter sentiment analyzer.

Table 6: Performance Comparison

	Rank in SemEval 2016	F1-score (%)
Switchcheese [18]	1	63.3
Sensei-LIF [7]	2	63.0
Unimelb [11]	3	61.7
Aueb [12]	5	60.5
Our system		63.7

6 CONCLUSIONS

This paper has introduced our approach to Twitter sentiment analysis. In the preprocessing step, we have proposed methods to deal with repeated characters in informal expression of words and contrast words in text. Different feature types have been carefully investigated and selected for the classification task. A soft-voting method has been proposed to combine results from three classifications (i.e., decision tree, kNN, and SVM). Experiment results shown that our proposed system achieved good results compared to related research in this field, using the same testing dataset. Future work include carrying out a more carefully investigation on the use of contrast words, as well as proposing new features using in classifying algorithms. Deep learning methods are also one of our research targets in order to improve the system performance of our sentiment analyzing system.

REFERENCES

[1] Corinna Cortes, Vladimir Vapnik, 1995. Support-Vector Networks, Machine Learning, 20, pp.273-297.

[2] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011). Sentiment analysis of twitter data. In Proceedings of the Workshop on Languages in Social Media, LSM '11, pp. 30{38, Portland, Oregon.

[3] Aisopos, F., Papadakis, G., Tserpes, K., & Varvarigou, T. (2012). Textual and contextual patterns for sentiment analysis over microblogs. In Proceedings of the 21st International Conference on World Wide Web Companion, WWW '12 Companion, pp. 453-454, New York, NY, USA.

[4] Bakliwal, A., Arora, P., Madhappan, S., Kapre, N., Singh, M., & Varma, V. (2012). Mining sentiments from tweets. In Proceedings of the 3rd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis, WASSA '12, pp. 11{18, Jeju, Republic of Korea.

[5] Ajay Deshwal, Sudhir Kumar Sharma. 2016. Twitter sentiment analysis using various classification algorithms. 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)

[6] Svetlana Kiritchenko, Xiaodan Zhu Xiaodan, Saif M. Mohammad. 2014. Sentiment Analysis of Short Informal Texts. Journal of Artificial Intelligence Research 50 (2014) 723-762

[7] Mickael Rouvier, Benoît Favre: SENSEI-LIF at SemEval-2016 Task 4: Polarity embedding fusion for robust sentiment analysis. *In Proceeding of NAACL-HLT 2016*, 202-208

[8] Duy Tin Vo and Yue Zhang. 2016. Don't Count, Predict! An Automatic Approach to Learning Sentiment Lexicons for Short Text. In Proceedings of ACL2016, 219-224.

[9] 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, pages 79–86.

[10] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, Veselin Stoyanov. 2016. SemEval-2016 Task 4: Sentiment Analysis in Twitter. In Proceeding of NAACL-HLT 2016.

[11] Steven Xu, Huizhi Liang, Timothy Baldwin: UNIMELB at SemEval-2016 Tasks 4A and 4B: An Ensemble of Neural Networks and a Word2Vec Based Model for Sentiment Classification. *In Proceeding of NAACL-HLT 2016*, 183-189

[12] Stavros Giorgis, Apostolos Rousas, John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2016. Aueb.twitter.sentiment at SemEval-2016 Task 4: A Weighted Ensemble of SVMs for Twitter Sentiment Analysis. *In Proceeding of NAACL-HLT 2016.*

[13] Hussam Hamdan. 2016. SentiSys at SemEval-2016 Task 4: Feature-Based System for Sentiment Analysis in Twitter. *In Proceeding of NAACL-HLT 2016*, 190-197.

[14] Mateusz Lango, Dariusz Brzezinski, Jerzy Stefanowski. PUT at SemEval-2016 Task 4: The ABC of Twitter Sentiment Analysis. 126-132. NAACL-HLT 2016.

[15] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov. 2016. Enriching Word Vectors with Subword Information. arXiv preprint arXiv:1607.04606

[16] Go, A., Bhayani, R., & Huang, L. 2009. Twitter sentiment classiffcation using distant supervision. Tech. rep., Stanford University.

[17] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Proceedings of the International Conference on Language Resources and Evaluation.

[18] Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurélien Lucchi, Valeria De Luca, Martin Jaggi: SwissCheese at SemEval-2016 Task 4: Sentiment Classification Using an Ensemble of Convolutional Neural Networks with Distant Supervision. *In Proceeding of NAACL-HLT 2016*, 1124-1128