A Sentiment Analyzer for Informal Text in Social Media

Huong Thanh Le [*]***, Nhan Trong Tran***
Hanoi University of Science and Technology - No. 1, Dai Co Viet, Hai Ba Trung, Hanoi, Viet Nam
Received: May 05, 2018; Accepted: November 26, 2018

Abstract

This paper introduces an approach to Twitter sentiment analysis, with the task of classifying tweets as positive, negative or neutral. In the preprocessing task, we propose a method 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, investigate and select an optimal classifying algorithm among Decision Tree, K Nearest Neighbor, Support Vector Machine, and a Voting Classifier for solving Twitter sentiment analysis problem. 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.

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

1. Introduction

Nowadays, 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. 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 working with traditional text because of the following reasons:

- Tweets are short. The size of a tweet is limited to 140 characters, which provides not enough information for classification algorithm working correctly.

- 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 these difficulties:

Example 1: *Ha-ha... I want to see. E macdonalds here cheaper. 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". Example 2 is harder to automatically analyze since it contains many informal words, "ya", "wans", “dunno", "4", "tmr", "anot", which are interpreted 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 reduce the system performance dramatically when applying traditional approaches in sentiment analysis. Several efforts have been made to solve this problem. Kiritchenko et al. [3] developed a linear-kernel SVM 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 [2] combined several feature types like emoticons, exclamation and question mark symbol, word gazetteer, unigrams and testing on six supervised classification algorithms.

Rouvier and Favre [4] used a CNN architecture for learning three polarity classifiers, each of which uses lexical, part-of-speech and sentiment words of the tweet as the input. A final fusion step was

[*] Corresponding author: Tel.: (+84) 904.674.102
Email: huonglt@soict.hust.edu.vn

applied, based on concatenating the hidden layers of the CNNs and training a deep neural network for the fusion. Aueb [6] used supervised learning with GloVe word embeddings for Twitter and weighted ensemble of classifiers. Lango et al. [8] 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 introduce our approach to Twitter sentiment analysis, with the task of classifying tweets as positive, negative or neutral, concentrating on reducing the effectiveness of the two problems mentioned above. A modified application of word embeddings is proposed to deal with informal expression and to compute semantic meaning of words. We investigate a method to deal with contrast words in determining sentence polarity. We propose features and investigate an optimal classification algorithms using these features to obtain the best outcome. Decision Tree (DT), K Nearest Neighbor (kNN), Support Vector Machine (SVM) are chosen as classification algorithms for 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 briefly describes word embeddings and our method of using word embeddings in our system. Section 3 introduces our approach to Twitter sentiment analysis. Our experimental results with different strategies to combine features are represented in Section 4. Section 5 concludes the paper and proposes directions for future work.

2. 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 [9] 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] [10] - a large Twitter dataset with many word extensions created by repeating some of its characters (e.g., "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 3.1 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.

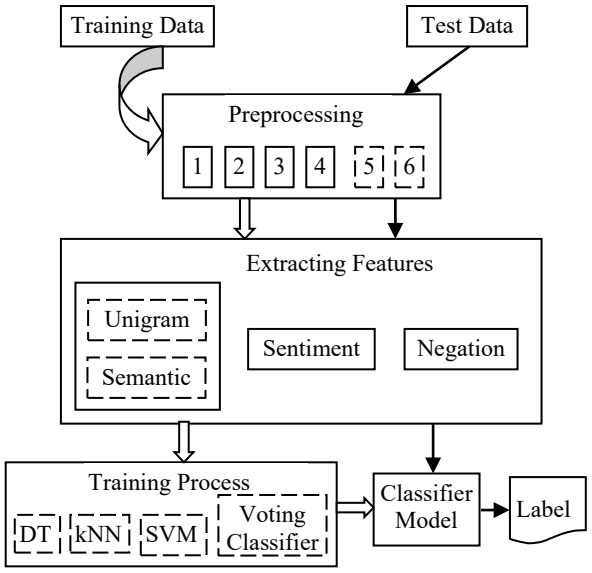

Fig. 1. Proposed system architectures

3. Proposed Twitter Sentiment Analyzer

The architecture of our proposed system is shown in Fig.1. Our system has been implemented with different scenario aiming at testing the effectiveness of our proposed preprocessing steps and finding the best classifying features. Numbers 1 to 6 in the preprocesing module correspond to six processing steps mentioned in Section 3.1, in which steps 5 and 6 are our proposed one. The boxes with dot lines in Extracting Features and Training Process modules indicate that only one of these boxes can be used in the given module at a time. Details of our testing scenario are discussed in Section 4.2.

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

The remained part of this section will discuss about our proposed preprocessing steps and features in details.

3.1 Preprocessing

As mentioned in Section 1, understanding tweets is challenging 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:

1. Lowercasing all the input text;
2. Converting all url to URL and @username to AT_USER;
3. Converting all abbreviations, slang and emoticons to their meaning (e.g., :) to “happy”, “dunno” to “don’t know”);
4. Removing all duplicate whitespace;
5. Replacing all three or more duplicate consecutive characters with two (e.g., niccccceeee to niccee).
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
wat	what
hee	here
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 frequently enough to be learned by the system. Beside the Twitter dictionary, word embedding model is also used in our system to get the actual meaning of slang and abbreviations.

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 [7] addressed this problem by using Brown corpus with 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 [4] solved the problem of informal expressions by using word embedding. 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

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

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.

Steps 5 and 6 are our new proposed pre-processing steps compared to other researches in this field. Therefore these steps will be tested carefully in our experiments, mentioned in Section 4.

3.2 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.

3.2.1 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., [7,8]) 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. There are 1,749,910 unigrams in our unigrams dictionary in total.

3.2.2 Semantic feature

Since tweets are very short and containing various modifications of words, representing tweets as vectors of unigrams as in some previous research (e.g., [7,8]) 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 2, 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.

3.2.3 Sentiment feature

The sentiment score of a tweet is calculated by summing word-sentiment associations of this tweet. SentiWordNet [11] 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.

3.2.4 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*" is a negative statement; "*Why am I feeling worse?*" is still a negative notion. Therefore, question mark is not used as a feature in our classification system.

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.

3.3. Classification algorithm

We consider the task of classifying a tweet as positive, negative, and neutral. Several classifying algorithms are tested in order to find the best performance one. K Nearest Neighbor and Support Vector Machines are chosen since they are widely used and provide high perfomance in this task. By empirical study different values of k, the number of neighbors (k) is set to 24, which gave us most accurate results. Besides, a Voting Classifier - a modifying version of Adaboost - is also used. This is a type of "Ensemble Learning" where multiple learners are employed to build a stronger learning algorithm. Since Decision Tree is often used as a default weak learner in Adaboost, it is also considered as a classifier in our experiments.

Our Voting Classifier applies 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 $\forall i$.

4. Experiments

4.1 Dataset

Three Twitters datasets were used in our experiments: Sentiment140, Twitter 2013 in SemEval2013 and Twitter 2016 in SemEval2016 for task 4, subtask A[§]. 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)	9,684	3,640	1,458	4,586
Twitter 2013 (dev)	1,654	575	340	739
Twitter 2016 (train)	6,000	3,094	863	2,043
Twitter 2016 (dev)	1,999	843	391	765
Our training data	19,337	8,152	3,052	8,133
Twitter 2016 (test)	20,632	7,059	3,231	10,342

4.2 Experimental Setting

Since all systems that we compared withused macro-averaged F1-score to evaluate the system performance, this measure was also used in our system. The first experiment was carried out to find the best algorithm among four classification algorithms mentioned in Section 3.3. Our proposed feature sets used in this experiment including semantic features, sentiment features, and negation feature. Table 4 presents our system performance withthese classifiers.

Table 4: Our System Performance with Four Classifiers

Classifier	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 the 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.

By analyzing system results, we found one 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 effectiveness of our proposed preprocessing steps 5 and 6, unigrams, semantic and negation features, we carried out experiments with our best classifier - Voting Classifier, using the following scenario:

1. using all preprocessing steps + unigrams + sentiment + negation features

2. using all preprocessing steps + semantic + sentiment + negation features

3. using all preprocessing steps + semantic + sentiment

4. using preprocessing steps 1,2,3,4,6 + semantic + sentiment + negation features

5. using preprocessing steps 1,2,3,4,5 + semantic + sentiment + negation features

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

Experimental results are shown in Table 5 below.

Table 5. Our System Performance with Different Feature Sets

Scenario	1	2	3	4	5
F1-score (%)	55.2	63.7	58.3	53.5	63.5

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. The F1-score in scenario 3 drops from 63.7% (in scenario 2) down to 58.3%, proving that negation feature is necessary for the sentiment analysis task.

To investigate the effectiveness of Step 5 in our preprocessing step, we removed this step from the preprocessing task; retrained Fasttext's word embedding model; retrained and tested the system with the new preprocessing module. The F1-score in this case fell dramatically from 63.7% to 53.5%. It proves that this step is very important in dealing with informal text as in social network.

The F1-score in scenario 5 reduces a little bit (0.2%) comparing to the case using contrast words. It indicates that using contrast words has a positive effect in this task. Analyzing system outputs points out 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 scenario gave us the best result of 63.7%, when using the Voting Classifier with the feature sets: semantic features, sentiment features, and negation feature.

5. 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. In this experiments, our system was compared with the top three systems at SemEval2016, which are Switchcheese [12], Sensei-LIF [4], and Unimelb [5]. We also compared our system with Aueb [6] and PUT [8]. Aueb achieved the highest result among the ones did not used deep learning at this contest. PUT [8] applied some boosting mechanisms (i.e., Random Forests, Gradient Boosting Trees) similar to us. However, it did not have the preprocessing steps 5 and 6 proposed by us. Note that each research used a different training set. Sensei-LIF [4] used the train and development corpora from Twitter 2013 to 2016 for training and Twitter 2016-dev as a development set. Aueb [6] trained the system by using data from SemEval-2013 Task 2 and SemEval-2016 Task 4. 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).

Table 6. Performance Comparison

	Rank in SemEval 2016	F1-score (%)
Switchcheese [12]	1	63.3
Sensei-LIF [4]	2	63.0
Unimelb [5]	3	61.7
Aueb [6]	5	60.5
PUT [8]	14	57.6
Our system		63.7

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 system outperforms the first ranked system in SemEval 2016 campaign using deep learning techniques. It proves that our preprocessing step 5 is very efficient in promoting the system performance. It boosts the F1-score of our system from a value lower than that of the 14th ranked system in SemEval 2016 to a value higher than that of the first ranked one (see Table 5 - scenario 2 and 4, and Table 6 for details).

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 voting classifier - a soft-voting method has been proposed to combine results from three classifications (i.e., Decision Tree, kNN, and SVM). Our experiment results show that our proposed system achieved good results compared to related research in this field, using the same testing dataset. Our 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] Ajay Deshwal,Sudhir Kumar Sharma. 2016. Twitter sentiment analysis using various classification algorithms. In Proceeding of CRITO 2016.

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

[4] 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

[5] 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

[6] 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.

[7] 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.

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

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

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

[11] 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.

[12] 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