ECNUCS: A Surface Information Based System Description of Sentiment Analysis in Twitter in the SemEval-2013 (Task 2)

Tian Tian ZHU and **Fang Xi ZHANG** and **Man LAN***
Department of Computer Science and Technology
East China Normal University
51111201046,51111201041@ecnu.edu.cn; mlan@cs.ecnu.edu.cn

Abstract

This paper briefly reports our submissions to the two subtasks of Semantic Analysis in Twitter task in SemEval 2013 (Task 2), i.e., the Contextual Polarity Disambiguation task (an expression-level task) and the Message Polarity Classification task (a message-level task). We extract features from surface information of tweets, i.e., content features, Micro-blogging features, emoticons, punctuation and sentiment lexicon, and adopt SVM to build classifier. For subtask A, our system on twitter data ranks 2 on unconstrained rank and on SMS data ranks 1 on unconstrained rank.

1 Introduction

Micro-blogging today has become a very popular communication tool among Internet users. Millions of messages are appearing daily in popular web sites that provide services for Micro-blogging and one popularly known is Twitter[1]. Through the twitter platform, users share either information or opinions about personalities, politicians, products, companies, events (Prentice and Huffman, 2008) etc. As a result of the rapidly increasing number of tweets, mining sentiments expressed in tweets has attracted more and more attention, which is also one of the basic analysis utility functions needed by various applications.

The task of Sentiment Analysis in Twitter is to identify the sentiment of tweets and get a better understanding of how sentiment is conveyed in tweets and texts, which consists of two sub-tasks, i.e., the Contextual Polarity Disambiguation task (an expression-level task) and the Message Polarity Classification task (a message-level task). The contextual polarity disambiguation task (subtask A) is to determine whether a given message containing a marked instance of a word or a phrase is positive, negative or neutral in that context. The message polarity classification task (subtask B) is to decide whether a given message is of positive, negative, or neutral sentiment and for messages conveying both a positive and negative sentiment, whichever is the stronger sentiment should be chosen (Wilson et al., 2013). We participate in these two tasks.

In recent years, many researchers have proposed methods to analyze sentiment in twitter. For example, (Pak and Paroubek, 2010) used a Part of Speech (POS) tagger on the tweets and found that some POS taggers can help identify the sentiment of tweets. They found that objective tweets often contain more nouns than subjective tweets. However, subjective tweets may carry more adjectives and adverbs than objective tweets. Besides, (Davidov et al., 2010) proved that emoticon and punctuation like exclamation mark are good features when distinguishing the sentiment of tweets. In addition, some sentiment lexicons like SentiWordNet (Baccianella et al., 2010) and MPQA Subjectivity Lexicon (Wilson et al., 2009) have been adopted to calculate the sentiment score of tweets (Zirn et al., 2011).

The rest of this paper is organized as follows. Section 2 describes our approach for subtask 1, i.e., the Contextual Polarity Disambiguation task. Section 3 describes our approach for subtask 2, i.e., the

[1] http://www.twitter.com

*Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 408–413, Atlanta, Georgia, June 14-15, 2013.

message polarity classification task. Concluding remarks is in Section 4.

2 System Description of Contextual Polarity Disambiguation

For the Contextual Polarity Disambiguation task, we first extract features from multiple aspects, i.e., punctuation, emoticons, POS tags, instance length and sentiment lexicon features. Then we adopt polynomial SVM to build classification models. According to the definition of this task, the given instance has been marked by a start position and an end position rather than a whole tweet. So we first record the frequency of the first three kinds of features in this given instance. To avoid interference from the number of words in given instance, we then normalize the feature values by the length of instance.

2.1 Preprocessing

Typically, most tweets contain informal language expressions, with creative spelling and punctuation, misspellings, slang, new words, URLs, and genre-specific terminology and abbreviations, such as, "RT" for "re-tweet" and #hashtags, which are a type of tagging for Twitter messages. Therefore, working with these informal text genres presents challenges for natural language processing beyond those typically encountered when working with more traditional text genres, such as newswire data. So we perform text preprocessing in order to remedy as many informal texts as possible. Firstly, we perform normalization to convert creative spelling and misspelling into its right spelling. For example, any repetition of more than 3 continuous letters are reduced back to 1 letter (e.g. "noooo" is reduced to "no"). In addition, according to the Internet slang dictionary[2], we convert each slang to its complete form, for example, "aka" is rewritten as "also known as". After that, we use the Stanford parser[3] for tokenization and the Stanford POS Tagger[4] for POS tagging. Finally, Natural Language Toolkit[5] is used for WordNet based Lemmatization.

[2]http://www.noslang.com

[3]http://nlp.stanford.edu/software/lex-parser.shtml

[4]http://nlp.stanford.edu/software/tagger.shtml

[5]http://nltk.org/

2.2 Features

2.2.1 Punctuation

Typically, punctuation may express user's sentiment to a certain extent. For example, many exclamation marks (!) in tweet may indicate strong feelings or high volume (shouting). Therefore, given a marked instance, we record the frequency of the following four types of punctuation: (1) exclamation mark (!), (2) question mark (?), (3) double or single quotation marks(" and "), (4) sum of the above three punctuation. Then the punctuation feature value is normalized by the length of instance.

2.2.2 Emoticons

We create two features that capture the number of positive and negative emoticons. Table 1 lists the two types of emoticons. We also use the union of the two emoticon sets as a feature. In total, we have three emoticon features.

Positive Emoticons	Negative Emoticons
:-) :) :D :-D =) ;)	:(:-(: (;(
;-) ;) ;D ;-D (; :)	;-(; ( ):
:-P ;-P XD (-: (-; :o) ;o)	-/ :/ ;-/ ;/
:0) ;0) ^_^	T_T T0T ToT

Table 1: List of emoticons

2.2.3 POS

According to the finding of (Pak and Paroubek, 2010), POS taggers help to identify the sentiment of tweets. Therefore, we record the frequency of the following four POS features, i.e., noun ("NN", "NNP", "NNS" and "NNPS" POS tags are grouped into noun feature), verb ("VB", "VBD", "VBG", "VBN", "VBP" and "VBZ" POS tags are grouped into verb feature), adjective ("JJ", "JJR" and "JJS" POS tags are grouped into adjective feature) and adverb ("RB", "RBR" and "RBS" POS tags are grouped into adverb feature). Then we normalize them by the length of given instance.

2.2.4 Sentiment lexicon Features

For each word in a given instance, we use three sentiment lexicons to identify its sentiment polarity and calculate its sentiment weight, i.e., SentiWordNet (Baccianella et al., 2010), MPQA Subjectivity Lexicon (Wilson et al., 2009) and an Unigram Lexicon made from the Large Movie Review Dataset

v1.0 (Maas et al., 2011). To calculate the sentiment score for this instance, we use the following formula to sum up the sentiment score of each word:

$$Senti(I) = \sum_{w \in I} \frac{Num(w) * Senti\_weight}{Length(I)} \quad (1)$$

where I represents the given instance and w represents each word in I. The $Senti\_weight$ is calculated based on the word in the instance and the chosen sentiment lexicon. That is, for each word in the instance, we have different $Senti\_weight$ values for it since we use different sentiment lexicons. Below we describe the calculation of $Senti\_weight$ values for a word in three sentiment lexicons. Note that $Num(w)$ is always 1 since most words appear one time in a instance.

SentiWordNet. SentiWordNet is a lexical resource for sentiment analysis, which assigns each synset of WordNet (Stark and Riesenfeld, 1998) three sentiment scores: positivity, negativity, objectivity (e.g. living#a#3, positivity: 0.5, negativity: 0.125, objectivity: 0.375), where sum of these three scores is always 1. For one concept, if its positive score and negative score are all 0, we treat it as objective concept; otherwise, we treat it as subjective concept. And we take the first sense as the concept of each word.

We extract three features from SentiWordNet, i.e., $SUB_{WordNet}$, $POS_{WordNet}$ and $NEG_{WordNet}$. The $Senti\_weight$ of $SUB_{WordNet}$ records whether a word is subjective. If it is subjective, we set $Senti\_weight$ as 1, otherwise 0. Similarly, the $Senti\_weight$ values of $POS_{WordNet}$ and $NEG_{WordNet}$ indicate the positive score and the negative score of the given word. Considering some negation terms may reverse the sentiment orientation of instance, we manually generate a negation term list (e.g. "not", "never", etc.,) and if a negation term appears in the instance, we switch the $POS_{WordNet}$ to $NEG_{WordNet}$ and vice versa. Besides, we adopt another feature to record the ratio of $POS_{WordNet}/NEG_{WordNet}$. If the denominator is 0, i.e., $NEG_{WordNet} = 0$, that means, the word has the strongest positive sentiment orientation, then we set 10*$POS_{WordNet}$ as its feature value.

MPQA. The MPQA Subjectivity Lexicon contains about $8,000$ subjective words. Each word in the lexicon has two types of sentiment strength: strong subjective and weak subjective, and four kinds of sentiment polarity: positive, negative, both (positive and negative) and neutral. Therefore we calculate three features from this lexicon, i.e., SUB_{MPQA}, POS_{MPQA} and NEG_{MPQA}. For the SUB_{MPQA} feature, if the word has strong or weak subjective, we set its $Senti\_weight$ as 1 or 0.5 accordingly. For the POS_{MPQA} (NEG_{MPQA}) feature, we set $Senti\_weight$ as 1, or 0.5 or 0 if the word has strong positive (negative), or weak positive (negative) or neutral. We also reverse the sentiment orientation of POS_{MPQA} and NEG_{MPQA} if a negation term appears.

Unigram Lexicon. Unlike the above two lexicons in themselves which provide sentiment polarity and sentiment strength for each word, we also utilize the third lexicon to calculate the sentiment information statistically. Therefore we generate an unigram lexicon by ourselves from a large Movie Review data set(Maas et al., 2011) which contains $25,000$ positive and $25,000$ negative movie reviews. We calculate the $Senti\_weight$ of each word appears in the data set as the ratio of the frequency of this word in positive reviews to that in negative reviews and record this feature as $Senti_{UL}$.

Clearly, since we use additional data set to develop a sentiment lexicon which is used to generate this $Senti_{UL}$ feature, this feature is worked with all other features to train the unconstrained system.

2.2.5 Other features

In addition, we collect three other features: (1) length of instance, (2) uppercase word (e.g. "WTO" or "Machine Learning"), (3) URL. For the uppercase word and URL features, we record the frequency of them and then normalize them by the instance length as well.

2.3 Experiment and Results

2.3.1 Classification Algorithm

We adopt LibSVM[6] to build polynomial kernel-based SVM classifiers. We have also tried linear kernel but get no improvement. To obtain the optimal parameters for SVM, such as c and g, we perform grid search with 10-fold cross validation on training

[6]http://www.csie.ntu.edu.tw/ cjlin/libsvm/

data.

2.3.2 Results and Discussion

In section 2, we obtained 22 features in total. To train the constrained model, we used the above described 21 features (except $Senti_{UL}$) and used all above 22 features to train the unconstrained model. We combined the provided training and development data by the organizers as our final training data. And we should apologize for our misunderstanding of the definitions of the constrained and unconstrained condition. As the official definition of unconstrained model, participates are allowed to add other data to expand the training data sets, but our unconstrained model only adds one feature ($Senti_{UL}$) which is got from other data set. Therefore, we actually submitted two results of constrained model. But we still refer this model trained on all features as unconstrained model for it appeared in the unconstrained list of official results. There are two kinds of test data: $4,435$ twitter instances and $2,334$ SMS message instances. Table 2 list the F-score and averaged F-score of positive, negative and neutral class of each test data set.

On one hand, from the table we can see that whether on constrained or unconstrained model, the results on twitter data are slightly better than those of SMS data. However, this difference is not significant. This indicates that the model trained on twitter data performs well on SMS data. And it also shows that twitter data and SMS data are linguistically similar with each other in nature. On the other hand, we find that on each test data set, there is little difference between the constrained model and the unconstrained model, which indicates the $Senti_{UL}$ feature does not have discriminating power by itself. However, since we had not used other labeled or unlabeled data to extend the training data set, we cannot draw a conclusion on this. Besides, our results contain no neutral items even though the classifier we used is multivariate. One reason may be the neutral instances in training data is too sparse for the classifier to learn.

On twitter data, our system ranks 2 under unconstrained model and ranks 10 under constrained model. On SMS data, our system ranks first under unconstrained model and ranks 7 under constrained model.

3 System Description of Message Polarity Classification

Unlike the previous subtask, the Message Polarity classification task focuses on the whole tweet rather than a marked sequence of given instance. Firstly, we perform text preprocessing as Task A. Besides the previous described features, we also extract following features.

3.1 Features

3.1.1 Micro-blogging features

We adopted three tweet domain-specific features, i.e., #hashtags, @USERS, URLs. We calculate the frequency of the three features and normalize them by the length of instance.

3.1.2 *n*-gram features

We used unigrams to capture the content of tweets.

3.2 Classification Algorithm

We adopted two different classifiers in preliminary experiments, i.e., maximum entropy and SVM. We used the Mallet tool (McCallum, 2002) to perform Maximum Entropy classification and LibSVM[7] with a linear kernel, where the default setting is adopted in all experiments.

3.3 Results on Training Data

In the first experiment, we used only content features and LibSVM classifier to do our experiments. The results were listed in Table 3. From Table 3, we found that the system with unigram without removing stop words performs the best. The possible reason was that Microblogs are always short (constrained in 140 words) and removing stop words would cause information missing in such a short text. In addition, although bigrams improved the performance to some extern, they added the feature space many more and might affect other features. So in our final systems, we used only unigram feature and did not remove stop words.

In the second experiment, we compared all features described before with two learning algorithms. The results were shown in Table 4, where 1 indicates unigram, 2 indicates micro-blog, 3 indicates

[7] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

System	F-pos	F-neg	F-neu	average F(pos and neg)
twitter-constrained	0.8506	0.7390	0.0	0.7948
twitter-unconstrained	0.8561	0.7468	0.0	**0.8015**
SMS-constrained	0.7727	0.7611	0.0	0.7669
SMS-unconstrained	0.7645	0.7824	0.0	**0.7734**

Table 2: Results of our systems on subtask A test data

features	F-pos	F-neg	F-neu	average F(pos and neg)	acc(%)
unigrams	**0.6356**	0.3381	**0.7122**	0.4869	**63.75**
unigrams(remove stop words)	0.6046	0.3453	0.6988	0.4750	62.13
bigrams	0.5186	0.0196	0.6625	0.2691	55.85
unigrams+bigrams	0.6234	**0.3724**	0.7043	**0.4979**	63.18

Table 3: Results of our systems on on subtask B training data using content features

features	F-pos		F-neg		F-neu		average F(pos and neg)		acc(%)	
	MaxEnt	SVM	MaxEnt	SVM	MaxEnt	SVM	MaxEnt	SVM	MaxEnt	SVM
1	0.6178	0.6356	0.3696	0.3381	0.6848	0.7122	0.4937	0.4869	61.56	63.75
1+2	0.6403	0.6339	0.4207	0.4310	0.6990	0.7184	0.5305	0.5324	63.75	64.89
1+2+3	0.6328	0.6512	0.4051	0.4371	0.6975	0.7232	0.5190	0.5442	63.18	65.75
1+2+3+4	**0.6488**	**0.6593**	**0.4587**	**0.4481**	**0.7083**	**0.7288**	**0.5538**	**0.5537**	**64.89**	**66.41**
2+3+4	0.5290	0.5201	0.2897	0.2643	0.6503	0.6411	0.4093	0.3922	55.85	54.80

Table 4: Results of our systems on subtask B training data using all features and two learning algorithms

punctuation, 4 indicates sentiment lexicon features. From Table 4, the best performance was obtained by using all these features. Since the performance of Maximum Entropy and SVM in terms of F-score was comparable to each other, we finally chose SVM since it achieved a better accuracy than MaxEnt.

3.4 Results on Test Data

We combined the provided training and development data by the organizers as our final training data. There were two kinds of test data: $3,813$ tweets and $2,094$ SMS messages . Table 5 listed the results of our final systems on the tweet and SMS data sets by using all above described features and SVM algorithm.

From Table 5, on one hand, we can see that the overall performance of SMS test data is inferior to twitter data, for the reason may be that the domain of features are all based on twitter data, and maybe not quite suitable for SMS data. However, this different is not significant. On the other hand, we also can find that there is no obvious distinction between the constrained and the unconstrained model on each test data.Also from Table 5, the F-score for positive instances is higher than negative instances, and it is interesting that most of other participants'systems results show the same consequence. One of the reason may be the positive instance in training data are more than negative instances both in training data and test data.

Our result on twitter message is 0.5842 , while on SMS is 0.5477. Compared with the highest average F-score 0.6902 in twitter data and 0.6848 in SMS data, our system does not perform very well. On the one hand , pre-processing was roughly , then features extracted were not suited in classification stage. On the other hand, in classification stage all parameters were default when used LibSVM. These might cause low performance. In future, we may overcome the insufficient described above and take hashtags' sentiment inclination and the source files of URLs into consideration to enhance the performance.

System	F-pos	F-neg	F-neu	average F(pos and neg)
twitter-constrained	0.6671	0.4338	0.7124	0.5505
twitter-unconstrained	**0.6775**	0.4908	0.7204	**0.5842**
SMS-constrained	0.5796	0.4846	**0.7801**	0.5321
SMS-unconstrained	0.5818	**0.5137**	0.7612	0.5477

Table 5: Results of our systems on subtask B test data

4 Conclusion

In this work we extracted features from four aspects, including surface information of twitters and sentiment lexicons like SentiWordNet and MPQA Lexicon. On the contextual polarity disambiguation task, our system ranks 2 on twitter (unconstrained) rank and ranks 1 on SMS (unconstrained) rank.

Acknowledgements

The authors would like to thank the organizers and reviewers for this interesting task and their helpful suggestions and comments, which improves the final version of this paper. This research is supported by grants from National Natural Science Foundation of China (No.60903093), Shanghai Pujiang Talent Program (No.09PJ1404500), Doctoral Fund of Ministry of Education of China (No. 20090076120029) and Shanghai Knowledge Service Platform Project (No. ZF1213).

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 241–249. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June. Association for Computational Linguistics.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of LREC*, volume 2010.

Sara Prentice and Ethan Huffman. 2008. Social medias new role in emergency management. *Idaho National Laboratory*, pages 1–5.

Michael M Stark and Richard F Riesenfeld. 1998. Wordnet: An electronic lexical database. In *Proceedings of 11th Eurographics Workshop on Rendering*. Citeseer.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational linguistics*, 35(3):399–433.

Theresa Wilson, Zornitsa Kozareva, Preslav Nakov, Alan Ritter, Sara Rosenthal, and Veselin Stoyanov. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Cäcilia Zirn, Mathias Niepert, Heiner Stuckenschmidt, and Michael Strube. 2011. Fine-grained sentiment analysis with structural features. In *Proceedings of the 5th international Joint conference on natural Language Processing (iJcnLP-2011)*, volume 167.