

# Twitter sentiment analysis with SVM and RNTN \*

Yun Wu      Qiren Chen      Xiaofan Lu

University of Texas at Austin

Computer Science Department

{ywu, qiren, xiaofan}@cs.utexas.edu

## Abstract

In this project, we address the issue of sentiment analysis of twitter message. We collected and preprocessed three types of corpus. Two models, Support Vector Machine(SVM) and Recursive Neural Tensor Network (RNTN) are evaluated by extensive experiments.

## 1 Introduction

Microblogging websites (such as Twitter, Weibo) have gained popularity in recent years. People can easily post real time, short message to express their opinions. Sentiment of microblog is of particular interest because such information is valuable in diverse areas such as entertainment, politics and economics.

However, sentiment analysis over twitter message is a challenging task due to the noisy nature of tweet. It contains ungrammatical sentences, typos, creative punctuations, slangs, new words, URLs, and genre-specific terminology and abbreviations, such as, RT for re-tweet, #hashtags, @mentions.

Most of existing works of sentiment analysis are based on bag of words classifiers. Different features to improve the performance of bag-of-words model have been proposed (Agarwal et al., 2011). Such classifiers can work well in longer documents by relying on a few words with strong sentiment such as "great" or "awesome". However, bag-of-words models have difficulties in handling negations

and comparisons, which rely on the structure of sentence.

Recursive Neural Tensor Network (RNTN) model is recently proposed to capture the compositional effects with higher accuracy. It represents a phrases through word vectors and a parse tree and then computes vectors for higher nodes in the tree using the same tensor-based composition function. (Socher et al., 2013). The authors claim that RNTN can accurately captures the sentiment change and scope of negation. However, the data set used in the above paper is a bunch of single sentences extracted from well formatted movie reviews, which is quite different from twitter message.

In this paper, we evaluate both SVM and RNTN model on twitter message to compare the performance of both models. We collect twitter message from different sources and labeled some of them. Such corpus has been pre-processed according to the natural of different models. Experiment results indicate that RNTN perform significantly better than SVM on well formatted movie reviews. When training on *Sentiment Treebank* and testing on general topic tweet, RNTN still performs slightly better. However, due the lack the annotated training set on twitter message, existing RNTN model fails outperform in SVM on the task of sentiment detecting on general topic tweet. In the end, we explore the possibility of domain adaptation over RNTN model.

The paper is organized as follows. Section 2 defines the task we are solving and explains our approach. Section 3 presents and discusses experiment results. Section 4 and Section 5 details related and future work. Section 6 concludes the paper.

---

This is the final project report for CS 388 Natural Language Processing at The University of Texas at Austin in 2014 Spring

## 2 Problem Definition and Algorithm

### 2.1 Task Definition

In this paper, we address the problem of sentiment analysis of Twitter message. To be more precise, given a message, we want to classify whether the message is of positive or negative (binary decision), or neutral sentiment (ternary decision). For messages conveying both a positive and negative sentiment, whichever is the stronger sentiment should be chosen. For the following two example messages, we are expected to return positive on the first one and negative on the second.

If u haven't seen #Rio2 yet-GO! You need to meet Gabi! Great singer. Cute. Absolutely hysterical! @KChenoweth pic.twitter.com/kkVBUKjqE3

The rio 2 has one of the worst soundtracks evvvvaaa. I'm at Alamo @Draft-house Cinema. @marissanicole11 http://4sq.com/1kKF8qE

This is an interesting task because sentiment of Twitter message can be used as a barometer for public mood and opinion in diverse areas such as entertainment, politics and economics. For example, it was used to provide information on the temporal dynamic of sentiment in reaction to the debate video between Barack Obama and John McCain (Diakopoulos and Shamma, 2010). There is also a report on "Berkshire Hathaway Stock Rises When Anne Hathaway Makes Headlines"<sup>1</sup>, which indicates that sentiment toward public figure may have potential influence over stock market.

However, twitter message presents greater challenges for sentiment analysis than traditional text genres, such as newswire data. Tweets are within 140 characters, often consists of a few short sentences or even a single sentence. The language used is very informal, 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.<sup>2</sup>

<sup>1</sup><http://goo.gl/WlfY1c/>

<sup>2</sup><http://www.cs.york.ac.uk/semeval-2013/task2/>

### 2.2 Algorithm Definition

We experiment with two different types of models: The first is the traditional bag-of-words model, namely SVM here. The other is the recently proposed Recursive Neural Tensor Network which works pretty good on movie reviews. We first introduce the pre-processing steps and then the models.

#### 2.2.1 Pre-processing

The Twitter language is known as informal and flexible. Such properties will provide information about sentiment and should be processed carefully. As the SVM and RNTN model are build upon two different basic ideas, we need to have different pre-processing step for each model. For SVM, our goal is to maximize the information about sentiment while reduce the sparsity of the feature vector. For RNTN, we try to formalize tweets to well-organized sentences such that Stanford parser can recognize it well.

**General features** First we replace all the remaining HTML entities like "&" to ASCII code "-". We also delete the quotes around the whole tweet. Then we transform all the URL address to the keyword "URL" and the user names(start with @) to the key word "target" to keep the structure of the sentences. Hashtags may contain sentimental information so we simply remove the "#" in the front of the hashtag. We transform all the lengthening words ("sooooo" instead of "so") to three repeated letters like "sooo". There are also many misspellings and slangs in Twitter and we use a slang dictionary to formalize it.

**Preprocessing for SVM** We extract additional features for SVM and attach "\_neg" to negated sentences. Detailed information can be found in Section 2.2.2. We also try to reduce feature space with word cluster (Owoputi et al., 2011). Firstly, we obtain hierarchical word clusters via Brown clustering. The algorithm partitions words into a base set of 1,000 clusters, and induces a hierarchy among those 1,000 clusters with a series of greedy agglomerative merges that heuristically optimize the likelihood of a hidden Markov model with a one-class-per-lexical-type constraint. Then we sort the word

```
001010110 never neva nvr nevr #never nev-
err nver neverrr nevaa neeeever nevet neeeever
nevva
111100010011 sad blessed frightening sad
lucky frustrating helpful impressed low steady
:((( d&d busy
```

### 2.2.2 SVM

1. Word unigram and bigram (Unigram or Bigram): We use three different methods to present unigram feature.

**Frequency:** The count of each word is used as unigram feature. For example, if good appears

**tf-idf:** It is a numerical statistic to reflect the importance of a word to a document. tf-idf is the product of two statistics, term frequency and inverse document frequency.

$$\begin{aligned} tfidf_{i,j} &= tf_{i,j} \times idf_i && \text{where} \\ tf_{i,j} &= \frac{n_{i,j}}{\sum_k n_{k,j}} \\ idf_i &= \log \frac{|D|}{|\{j: t_j \in d_i\}|} \end{aligned}$$

2. Elongated word (**-elongated**): Lengthening words by repeating letters (such as coooooo, goooooo) is reported to be an strong indication of sentiment. Brody et al. (2011) reported that on average, one out of six sentences has word lengthening. In our experiment, we also take elongated word as a feature.
3. Targets, Hashtags, URLs, Numbers (**-url-num**): These are the features related to Twitter. Users of Twitter use the symbol to refer to other users in this manner automatically alerts them. Users use hashtags # to mark topics.
4. Negation (**-neg**): The effect of negation word is not simply the opposite of the original sentiment. In this paper, we try to append a NEG suffix to every word appearing between a negation and a clause-level punctuation mark.

In Recursive Neural Tensor Network, each word is represented as a  $d$ -dimensional vector. When an  $n$ -gram is given to the compositional models, it is parsed into a binary tree (as in Figure 1). We compute the parent vector in a bottom up fashion using a compositionally function  $g$  and use node vectors as features for a classifier at that node.

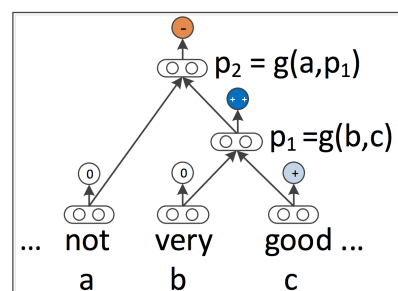


Figure 1: Trigram Example (Socher et al., 2013)

RNTNs use the following equations to compute the parent vectors:

$$p_1 = f \left( \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix} \right)$$

where  $f = \tanh$  is standard element-wise non-linearity.  $V^{[1:d] \in \mathbb{R}^{2d \times 2d \times d}}$  is the tensor that defines multiple bilinear forms.  $W \in \mathbb{R}^{d \times 2d}$  is the main parameter to learn.

The next parent vector  $p_2$  in the tri-gram will be computed with the same weights:

$$p_2 = f \left( \begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix} \right)$$

As we use the RNTN model as a black box in this project, so I skip the details on how to train the model. Interested reader could refer to the paper by Socher et al., (2013).

### 3 Experiment

#### 3.1 Corpus

We conduct our experiments on three types of corpus.

- The first corpus is the Stanford sentiment treebank released by Socher et. al. (2013). It is based on the dataset introduced by Pang and Lee (2005) and consists of 11,855 single sentences extracted from movie reviews on <http://www.rottentomatoes.com/>. It was parsed with the Stanford parser (Klein and Manning, 2003) and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges. To the best of our knowledge, it is the only public available corpus upon which a RNTN sentiment model can be trained right now. We refer to this corpus by *Sentiment Treebank* in the reset of paper.
- The second corpus is movie reviews on Twitter. We have 364 tweets extracted from two specialized review accounts (@FilmReviewIn140, @MovieTwoosh). Such reviews are mostly well formatted, usually consist of several sentences. The author rated each movie with A to F grades and the label the sentiment of the

tweet base on the grade. Tweets with a grade no worse than B are labeled positive otherwise negative. We refer to this corpus by *moive* in the reset of paper.

- The third corpus is general tweet message. It is taken from SemEval-2013: Sentiment Analysis in Twitter Task B<sup>3</sup>. In their release, each of the tweet messages has been manually labeled as positive, negative, or neutral. Out of all the 5,750 messages, 2,042 are positive, 855 are negative and 2853 are neutral. We refer to this corpus by *SemEval* in the reset of paper.

#### 3.2 Single Sentence Sentiment

We firstly evaluate both models using *Sentiment Treebank*, The same training/testing splits as in the original paper(Socher et al., 2013) is used.

Model	Accuracy		
	positive	negative	overall
RNTN	80.83	87.91	84.27
SVM <sub>S</sub>	74.15	73.79	73.97
SVM <sub>L</sub>	75.90	73.90	74.90

Table 1: Binary decision

In SVM<sub>S</sub>, we use a smaller dictionary (1635 words) in which words appear more than 10 times are used. In SVM<sub>L</sub>, we build a larger dictionary (3504 words) with the bar lowered to 5 times.

As we can see from Table 1, the performance of RNTN model is significantly better than SVM models. This is consistent with the result of the original paper(Socher et al., 2013). The huge boost comes from the fact that the structure of the sentence is utilized in the RNTN model. It might seems to be an unfair comparison as more information is needed (the sentiment of each parse of the sentence) in training the RNTN model. However, even if we train the SVM model with sentiment label of each parse as well, the result won't improve much as what SVM sees is just a subset of the original bag.

#### 3.3 Multiple Sentences Sentiment

We then evaluate how to combine the sentiment of multiple sentences. This is not an issue of SVM be-

<sup>3</sup><http://www.cs.york.ac.uk/semeval-2013/task2/index.php?id=data>

cause only a larger bag is needed. However, RNTN relies on the structural of single sentence so we need to combine the sentiment from multiple sentences within a single tweet. Here, we train the model on *Sentiment Treebank* and test it on *movie* corpus.

As for the RNTN model, we evaluate two ways to combine the sentiment of the whole tweet (multiple sentences). The first is to make hard (binary) decision on single sentence (either positive or negative) and sentiment of the tweet is decided by majority vote. Soft information (probability) is only used to break a tie. The second way fully relies on soft (probability) information. For each sentence, we generate a 5-element vector for the probability of the having the corresponding sentiment (very negative, negative, neutral, positive, very positive). We add the vector for all the sentences together and make final decision based on the combined vector.

Model	Accuracy(%)		
	positive	negative	overall
RNTN <sub>hard</sub>	70.08	81.54	74.18
RNTN <sub>soft</sub>	78.21	80.0	78.85
SVM	73.50	66.92	71.15

Table 2: Binary decision

From Table 2, we can see that both RNTN models perform better than SVM. For the two RNTN models, hard decision model has worse performance than soft decision in positive sentiment but slightly better on negative sentiment. This indicates that RNTN model tend to label a sentence as negative. Soft combining outperforms hard combining by more than 4% in the overall result. This is reasonable because more information is available in soft decision combining. Based on the above result, we use soft mode in the following experiments.

### 3.4 General topic Tweet

We conducted three sets of experiment over the general topic twitter corpus *SemEval*.

- Exp 1: Training on 90% of *SemEval* and testing on the rest 10%. This is the classic experiment setup. However, as we don't have annotated sentiment parse tree (the sentiment of each node of the parse tree, which is required to train the RNTN model), a RNTN model based

on *SemEval* can not be trained. Thus, we evaluate the following features of SVM model in this experiment.

Feature		Accuracy (%)
Unigram	Binary feature	78.84
	Frequency	78.64
	Tf-idf	78.22
Bigram		70.41
Unigram-Bigram		78.26
Unigram-url-num		<b>79.32</b>
Unigram-elongated		79.01
Unigram-url-num-elongated		79.19
Unigram-neg		78.77
Unigram-neg-url-num		78.19
Unigram-neg-enlongated		78.34
Unigram-neg-url-num-enlongated		77.95

Table 3: Experiment 1

- Exp 2: Training on *Sentiment Treebank* and testing on *SemEval*.

In this experiment, we apply the model trained on *Sentiment Treebank* on the general topic twitter message. It may sounds wired but as we mentioned earlier, *Sentiment Treebank* is the only corpus upon which we can train RNTN model. To be fair, we train a SVM model in the same way and compare the performance of the two in both binary-decision and ternary-decision task.

Model	Accuracy(%)		
	positive	negative	overall
RNTN	69.83	70.17	69.93
SVM	67.14	60.91	65.30

Table 4: Experiment 2.1

Model	Accuracy(%)			
	positive	negative	neutral	overall
RNTN	48.12	43.17	49.40	48.02
SVM	52.52	23.86	44.69	37.14

Table 5: Experiment 2.2

From Table 4 and Table 5, we can see that RNTN out perform SVM in both binary decision and ternary decision. However, the performance is worse than the result from Table 3.

The huge decrease in SVM model resulted from the difference of the two corpus. Clearly, twitter specific feature helps SVM model a lot but such feature is not available in the *Sentiment Treebank*.

Another interesting comparison is the performance of RNTN in Table 1 and Table 5. Like statistical parser, RNTN model is also quite specific to the genre of the training corpus. To make the problem worse, the Stanford parse doesn't work quite well on twitter message due to the noisy nature of it.

- Exp 3: Domain adaptation of RNTN

In this experiment, we explore domain adaptation of RNTN sentiment model. This is an interesting experiment as gathering fully annotated training data on twitter message is expensive and labor intensive. We have fully annotated corpus in *Sentiment Treebank* but *SemEval* is only been labeled positive, neutral or negative. In this experiment, we first train the model on *Sentiment Treebank*, then use the above model to label the training set of *SemEval* in sentiment treebank format. We re-train the model with both *Sentiment Treebank* and the labeled training set of *SemEval*. In the end, we test the new model on the test set of *SemEval*.

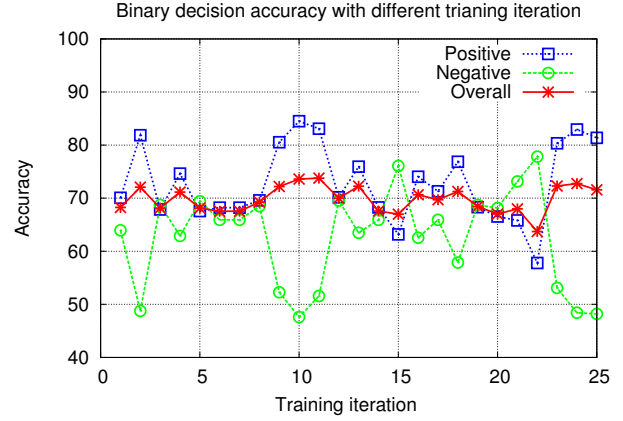


Figure 2: Binary decision

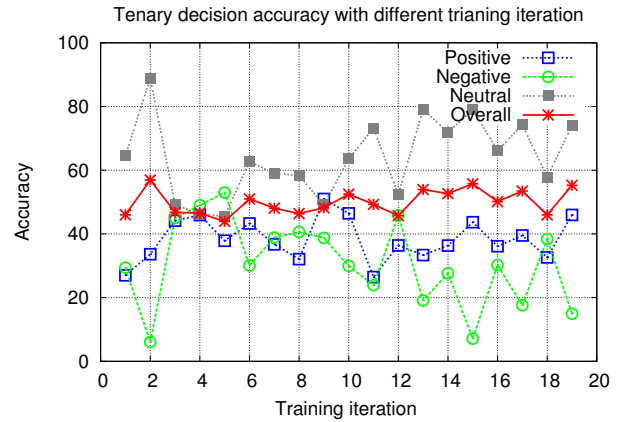


Figure 3: Ternary decision

We train the model with different iterations and evaluate all the models. As we can see from the above two figures, the result fluctuates a lot. Unfortunately, we don't see significant improvement in performance with domain adaptation.

### 3.5 Discussion

#### 3.5.1 SVM

We explore the usage of n-gram combined with other features.

- **Unigram:** Experiment results show that use binary format to present unigram performs best.

The result of using frequency and Tf-idf to present unigram is slightly worse than using binary format, but almost the same. The reason for this is Tweet is rather short (at most 140 characters). One word is hardly to appear twice in one Tweet. So frequency is not an obviously good feature in Tweets as in other corpus. For Tf-idf, because there are so many topics in Tweets, each word is hardly used to distinguish sentiment.

- **Bigram:** We use bigrams to help with tweets that contain negated phrases like not good or not cool. In our experiments, Bigram as a feature does not improve accuracy. However, bigrams tend to be very sparse and the size of our corpus is relatively small. So bigram doesn't work well on Twitter sentiment analysis. As we can see, the overall accuracy drops. In general using bigrams as features is not helpful because the feature space is very sparse. Thus we want to combine Unigram and Bigram as features.
- **Unigram-Bigram:** Results show Unigram-Bigram as a feature doesn't help. Compared to Unigram features, accuracy drops from 78.84% to 78.26%. For Alec Go (2009), there was also a decline for Unigram-Bigram as features.
- **Targets, Hashtags, URLs, Numbers:** The feature combining URL, Numbers with Unigram performs best in our experiment. The result shows that the number of URLs and Numbers is helpful for this task.
- **Elongated:** Elongated word as a feature doesn't help. The reason is that we can hardly classify different elongated forms of one word into one classification, such as goood and gooooodddd. So elongated words as a feature are sparse and hardly provide other information. In the future work, if we have a better algorithm to deal with elongated words, we can improve the results.
- **Negations:** Negations as features work almost as well as Unigram as a feature. The reason why negations don't improve the results is that it makes the dictionary larger than the original one. But since the size of corpus is relatively

small, so negations don't show its advantage in our experiment.

## 4 Related Work

## 5 Future work

- **Better pre-processing technique**  
Due to the noisy nature of twitter message, how well we can preprocess it matters a lot. How to preserve the structure of the original sentence and filtering out unnecessary information is a challenging task.
- **Better parser for twitter message**  
As the RNTN is built upon parse tree, the performance of the parser is of great importance to RNTN model. However, most of existing parser, including the one we used (Stanford Parser) are trained on newswire data, which is very different from the twitter message. We believe within a few years, the focus of the research community on twitter message will shift from current POS tagging to building accurate parser.
- **Sentiment treebank over twitter message**  
Our experiments are limited by the fact that there is no labeled sentiment treebank of twitter message. This is largely because that RNTN model has just come out that it has not been touched by the research community. Building our own sentiment treebank would be too much for us as a course project. However, it would be of great help if such a sentiment treebank over twitter message is available to the research community.

## 6 Conclusion

In this project, we evaluate both SVM and RNTN model on the task of twitter sentiment analysis. With the same setup, RNTN model performs better than SVM model. However, due to the lack of the annotated corpus (parse tree with sentiment label) over twitter message, RNTN model has worse performance than SVM model on sentiment decision over general topic tweet. Domain adaptation of RNTN model from movie review to twitter message gives no significant improvement.

## Acknowledgments

We would like to thank Dr. Mooney and our TA Zihao Zhou for their guidance and help on this course project.

## References

- [Socher et al.2013] , Socher, Richard and Perelygin, Alex and Wu, Jean and Chuang, Jason and Manning, Christopher D. and Ng, Andrew Y. and Potts, Christopher. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebanka *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*
- [Diakopoulos and Shamma 2010] , N. Diakopoulos, D. A. Shamma. April, 2010. Characterizing Debate Performance via Aggregated Twitter Sentiment. *Conference on Human Factors in Computing Systems (CHI)*.
- [Agarwal et al.2011] , Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of Twitter data. *In Proceedings of the Workshop on Languages in Social Media (LSM '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 30-38.
- [Owoputi et al.2011] , Olutobi Owoputi and Chris Dyer and Kevin Gimpel and Nathan Schneider and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters *In Proceedings of NAACL*