# ECNU at SemEval-2017 Task 5: An Ensemble of Regression Algorithms with Effective Features for Fine-Grained Sentiment Analysis in Financial Domain

**Mengxiao Jiang**[1], **Man Lan**[1,2*], **Yuanbin Wu**[1,2]
[1]Department of Computer Science and Technology,
East China Normal University, Shanghai 200062, P.R.China
[2]Shanghai Key Laboratory of Multidimensional Information Processing
51151201080@stu.ecnu.edu.cn, mlan,ybwu@cs.ecnu.edu.cn

## Abstract

This paper describes our systems submitted to the Fine-Grained Sentiment Analysis on Financial Microblogs and News task (i.e., Task 5) in SemEval-2017. This task includes two subtasks in microblogs and news headline domain respectively. To settle this problem, we extract four types of effective features, including linguistic features, sentiment lexicon features, domain-specific features and word embedding features. Then we employ these features to construct models by using ensemble regression algorithms. Our submissions rank 1st and rank 5th in subtask 1 and subtask 2 respectively.

## 1 Introduction

SemEval-2017 Task 5 is Fine-Grained Sentiment Analysis on Financial Microblogs and News(Cortis et al., 2017), focusing on identifying positive (bullish; believing that the stock price will increase) and negative (bearish; believing that the stock price will decline) sentiment associated with stocks and companies from microblogs and news domains. Unlike previous sentiment analysis, in this task, the fine-grained sentiment analysis not only contains sentiment orientation (i.e., positive or negative of the sentiment score) but also sentiment strength (i.e., the value of the sentiment score) attached to a particular company or stock explicitly or implicitly expressed in given texts.

Given a text instance (a microblog message from Twitter or StockTwits in subtask 1, a news statement or a headline in subtask 2), the goal of participants is to predict the sentiment score for each of the stocks and companies mentioned. The sentiment score is a floating value in the range of -1 (very negative) to 1 (very positive), with 0 designating neutral sentiment. Each microblog instance contains the following 5 items: "id", "source" (i.e., Twitter or StockTwits), "cashtag" (i.e., the company stock symbols to be predicted, e.g. *"$AAPL"*), "spans" and "sentiment score". And each news headline instance contains 4 items: "id", "company" (i.e., the company to be predicted), "title" and "sentiment score".

There are several differences between the spans in subtask 1 and the title in subtask 2: (1) The spans are sentence fragments related to the cashtag to be predicted, whereas the title is a complete sentence; (2) The spans almost regard one cashtag while the title usually contains one or more companies; (3) Due to (1) and (2), the spans contain less words but more effective information and less noises, which is contrary to the title.

In this work, the similar method is adopted for two subtasks. We extract a series of elaborately designed features. In addition to linguistic features, sentiment lexicon features and word embedding features, we also extract some domain-specific features for this task. Besides, we examine multiple different regression algorithms and ensemble methods are used to improve the performance of our models.

The rest of this paper is structured as follows. Section 2 describes our system in details, including data preprocessing, feature engineering, learning algorithms and evaluation measure. Section 3 reports datasets, experiments and results discussion. Finally, Section 4 concludes our work.

## 2 System Description

To solve these two subtasks, we extract lots of traditional NLP features combined with multiple machine learning algorithms to build supervised regression models. Due to the differences of data forms and data sources between the two subtasks, we adopt different features and algorithms



*Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 888–893,
Vancouver, Canada, August 3 - 4, 2017. 

for each subtask.

Specifically, for subtask 1, we rebuild the metadata of training and test set respectively with the official API of Twitter and StockTwits. The metadata contains the following information: tweets info or StockTwits info (e.g., "retweet_count"), users info (e.g., "favourites_count") and entities info (e.g., "sentiment"). As most of metadata of Twitter in test dataset is missing, we only extract the metadata features for StockTwits.

### 2.1 Data Preprocessing

Since the differences between the spans and the title described in section 1, for subtask 2, we replace the target company with *"TCOMPANY"* and replace other company with *"OCOMPNAY"* in the title.

For both subtasks, the subsequent preprocessing is the same. We firstly replace all URLs with "url" and transform the abbreviations, punctuation with a special format, slangs and elongated words to their normal format. Then, we use *Stanford CoreNLP tools*(Manning et al., 2014) for tokenization, POS tagging, named entity recognizing (NER) and parsing. Finally, the WordNet-based Lemmatizer implemented in NLTK[1] is adopted to lemmatize words to their base forms with the aid of their POS tags. And the word stemmer based on the Porter stemming algorithm and implemented in NLTK is adopted to remove morphological affixes from lemmatized words.

### 2.2 Feature Engineering

We extract the following four types of features to construct supervised regression models for two subtasks, i.e., linguistic features, sentiment lexicon features, domain-specific features and word embedding features.

#### 2.2.1 Linguistic Features

***N-grams*:** We remove the cashtag, punctuation, words that contain numbers and words with a length less than 2 from the sentence, and then extract 3 types of Bag-of-Words features as *N-grams* features, where N = {1,2,3} (i.e., *unigram*, *bigram*, and *trigram* features).

***RF N-grams*:** Differ from the *N-grams* features where each token shares the same weight, we calculate the weight for each token similar to (Lan et al., 2009). We firstly count the number of occurrences of each token in the positive and negative samples in the training data. Then we calculate the weight (i.e., *rf*) for each token in *unigram*, *bigram* and *trigram* as follows:

$$rf = max\left(ln(2 + \frac{a}{max(1,c)}),\ ln(2 + \frac{c}{max(1,a)})\right)$$

where $a$ is the number of sentences in the positive category that contain this token and $c$ is the number of sentences in the negative category that contain this token.

Finally, using a method similar to the *N-grams* features, we extract 3 types of *RF N-grams* features, where N = {1,2,3}. The difference between these two features is that *RF N-grams* features use the corresponding *rf* weight whereas *N-grams* features use 1 to represent the occurrence of words.

***Verb*:** Verbs usually contain more subjective tendencies. Thus, we also extract verbs (whose corresponding POS tags are VB, VBD, VBG, VBN, VBP and VBZ) from the sentence as *Verb* features with the Bag-of-Words form.

***NER*:** Considering that the money, number and percent informations might be useful for predicting the sentiment score of stocks in financial domain, we extract 11 types of named entities (i.e., DATE, DURATION, LOCATION, MONEY, NUMBER, ORDINAL, ORGANIZATION, PERCENT, PERSON, SET, TIME) from the sentence and represent each type of named entity as a binary feature to check whether it appears in the current sentence.

***Word Cluster*:** Since the high dimension of *N-grams* features, we also extract *word cluster* features to reduce the dimension of sentence representation (compared with *N-grams* features).

The *word cluster* features are extracted as follows: Firstly, we used the publicly available *Google word2vec*[2][3] that were trained on 100 billion words from Google News with the Skip-gram model (Mikolov et al., 2013) to get a 300-dimensional vector for each word in sentence. Then we use the *K*-means algorithm (k = 50) to cluster the words in the 300-dimensional vector space, and the value of k is chosen according to the preliminary experiment. After that, the word in sentence is replaced by its corresponding cluster assignment to get *word cluster* features.

[1] http://nltk.org

[2] https://code.google.com/archive/p/word2vec

[3] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

#### 2.2.2 Sentiment Lexicon Features

We also extract sentiment lexicon features (*SentiLexi*) to capture the sentiment information of the given sentence.

For each word in the sentence, we calculate five sentiment scores for each sentiment lexicon to construct *SentiLexi*: (1) the ratio of positive words, (2) the ratio of negative words, (3) the maximum sentiment score, (4) the minimum sentiment score, (5) the sum of sentiment scores. We transform the sentiment scores in all sentiment lexicons to the range of $[-1, 1]$, where "$-$" denotes negative sentiment. If the word does not exist in one sentiment lexicon, its corresponding score is set to zero. The following 8 sentiment lexicons are adopted in our systems: *Bing Liu opinion lexicon*[4], *General Inquirer lexicon*[5], *IMDB* (Zhu et al., 2013), *MPQA*[6], *AFINN*[7], *SentiWordNet*[8], *NRC Hashtag Sentiment Lexicon*[9], *NRC Sentiment140 Lexicon*[10].

#### 2.2.3 Domain-specific Features

Observing data, we found that the data in financial domain usually contains numbers. These numbers can indicate the degree of bullish or bearish, which has an important impact on the sentiment score of stocks or companies in financial domain. Moreover, we found that "call" and "put" are terminologies usually used in microblog domain and related to sentiment score. Therefore, we design the following domain-specific features.

***Number*:** We design 14 binary features to indicate whether there are the following types of numbers in the sentence: (1) *+num* (with a "+" in front of the number, e.g., "+5" ); (2) *-num*; (3) *num%*; (4) *+num%*; (5) *-num%*; (6) *$num*; (7) *num_word* (the number mixed with characters, e.g., "5am"); (8) *ordinal number* (e.g., "2nd"); (9) *num-num*; (10) *num-num%*; (11) *num-num-num*; (12) *num/num*; (13) *num/num/num*; (14) *only number* (there are no symbols and characters before and after the number).

***Keyword+Number*:** Based on the *Number* features, we defined 4-dimensional *Keyword+Number* features to indicate whether there is "call" (or "calls", "called") or "put" (or "puts") before "*+num%*" or "*-num%*" in the sentence.

Metadata usually contains information on the user who posted this tweet or StockTwits. The user information is useful, because it reflects the degree of authority or confidence of the posed tweet and StockTwits. Moreover, it also includes some extra useful informations about this tweet or StockTwits. Therefore, we extract the following *Metadata* features.

***Metadata*:** As most of metadata in Twitter is missing in test dataset, we used 8 items of the metadata in StockTwits to design following three types of features: (1) *Binary features* include the following items corresponding to the key in the metadata (json format): "source", "user/official", 'entities/sentiment", "liked_by_self" and "conversation/parent". (2) *Value features* contain the values of "conversation/replies" and "likes/total". And the *Value features* are standardized using [0-1] normalization. (3) *Other features*: "created_at" indicates whether the StockTwits is created in [0am, 9am), [9am, 3pm) or [3pm, 24pm). In total, we obtain 12 features from metadata.

***Punctuation (Punc)*:** People often use exclamation mark(!) and question mark(?) to express surprise or emphasis. Therefore, we extract the following 6 features: (1) whether there is "!" in sentence; (2) whether there is "?" in sentence; (3) the number of "!" in sentence; (4) the number of "?" in sentence; (5) the number of "$" in sentence; (6) the number of continuous "!" and "?" in sentence, e.g., "!!!", "????" or "!!??".

#### 2.2.4 Word Embedding Features

The previous work (Zhang and Lan, 2016; Jiang et al., 2016) on sentiment analysis task has proved the effectiveness of word embedding features. In this part, we utilize the *Google word2vec* to get the representation of the sentence.

***GoogleW2V*:** Unlike the *word cluster* features, the *Google word2vec* features (*GoogleW2V*) are extracted as follows: We firstly use the *Google word2vec* to get a 300-dimensional vector for each word in sentence. Then, the simple *min*, *max*, *average* pooling strategies are adopted to concatenate sentence vector representations with dimensionality of 900.

### 2.3 Learning Algorithms

For both tasks, we explore 7 algorithms as follows: AdaBoost Regressor (ABR), Bagging Regressor

[4]http://www.cs.uic.edu/liub/FBS/sentiment-analysis.html
[5]http://www.wjh.harvard.edu/inquirer/homecat.htm
[6]http://mpqa.cs.pitt.edu/
[7]http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010
[8]http://sentiwordnet.isti.cnr.it/
[9]http://www.umiacs.umd.edu/saif/WebDocs/NRC-Hashtag-Sentiment-Lexicon-v0.1.zip
[10]http://help.sentiment140.com/for-students/

(BR), Random Forest (RF), Gradient Boosting Regressor (GBR) and LASSO implemented in scikit-learn toolkit(Pedregosa et al., 2011), Support Vector Regression (SVR) implemented in liblinear toolkit(Fan et al., 2008) and XGBoost Regressor (XGB)[11] provided in (Friedman, 2001). All these algorithms are used with default parameters.

### 2.4 Evaluation Measure

To evaluate the performance of different systems, the official evaluation measure *weighted cosine similarity* (WCS) is adopted for two subtasks. Cosine similarity and cosine_weight will be calculated according the equation 1 and 2 respectively, where $G$ is the vector of gold standard scores and $P$ is the vector of scores predicted by the system. The final score is the product of the cosine and the weight (i.e., $WCS = cosine\_weight * cosine(G, P)$).

$$cosine(G, P) = \frac{\sum_{i=1}^{n} G_i \times P_i}{\sqrt{\sum_{i=1}^{n} G_i^2} \times \sqrt{\sum_{i=1}^{n} P_i^2}} \quad (1)$$

$$cosine\_weight = \frac{|P|}{|G|} \quad (2)$$

## 3 Experiment

### 3.1 Datasets

We conduct the experiments on the official datasets constructed by SSIX project (Davis et al., 2016), which consist of microblog messages (from Twitter or StockTwits) in subtask 1 and news headlines in subtask 2. Table 1 shows the statistics of the datasets used in our experiments.

| Domain | | Dataset | Instance | Metadata | Positive | Negative | Neutral |
|---|---|---|---|---|---|---|---|
| Microblog ( subtask 1) | Twitter | train | 765 | 603 | 246 | 510 | 9 |
| | | test | 371 | 6 | 116 | 243 | 6 |
| | StockTwits | train | 934 | 926 | 330 | 586 | 18 |
| | | test | 429 | 423 | 141 | 280 | 8 |
| Headline ( subtask 2) | | train | 1156 | - | 658 | 460 | 38 |
| | | test | 491 | - | 276 | 203 | 12 |

Table 1: Statistics of training and test datasets of two subtasks. *Positive, Negative and Neural* stand for the number of corresponding instances whose sentiment score is positive, negative and zero.

### 3.2 Experiments on Training Data

#### 3.2.1 Comparison of Different Algorithms

Table 2 shows the results of different algorithms using all features described before. Note that we did not use the *Metadata* feature in subtask 2 as there is no metadata in news headline domain. The 5-fold cross validation is performed for system development.

| Method | Algorithm | Subtask 1 | Subtask 2 |
|---|---|---|---|
| Single | SVR | 0.7450 | 0.7078 |
| | XGB | 0.7412 | 0.5875 |
| | ABR | 0.7382 | 0.6310 |
| | BR | 0.7441 | 0.5655 |
| | RF | 0.7373 | 0.5856 |
| | GBR | 0.7358 | 0.6763 |
| | LASSO | 0.3344 | 0.3297 |
| Ensemble | SVR + GBR | 0.7679 | **0.7231** |
| | SVR + XGB + ABR + BR | **0.7827** | 0.6875 |

Table 2: Results of algorithm selection experiments for two subtasks in terms of *WCS* on training datasets.

From Table 2, we find that for both subtasks, SVR outperforms other algorithms and LASSO performs the worst among all algorithms. Other algorithms perform differently on two subtasks. Therefore, we also perform experiments using an ensemble method. The last two rows in Table 2 list the results of using the top two and top four algorithms to build the ensemble regression models (named EN(2) and EN(4)), which average the output scores of all regression algorithm. From Table 2, we find that the ensemble classifier greatly increased the performance on both subtasks. Specifically, for subtask 1, the ensemble with top 4 algorithms improve 4% and for subtask 2, ensemble with top 2 improved 2% compared with the top score using a single regression algorithm. Therefore, we chose the EN(4) for subtask 1 and EN(2) for subtask 2 as the regression algorithm in following experiments.

#### 3.2.2 Feature Selection

Table 3 shows the best feature sets for two subtasks. Based on previous ensemble algorithms, we adopt *hill climbing* algorithm to select best features. That is, keep adding one type of feature at a time until no further improvement can be achieved. From Table 3, we find that: (1) The *RF N-grams*, *Verb*, *Word Cluster*, *SentiLexi*, *Number*, *Punctuation* and *GoogleW2V* features are beneficial for both subtasks; (2) Specially, the *NER* features and *Keyword+Number* features are more effective in subtask 1 than subtask 2.

To further analysis the significance of different features, we conduct the ablation experiments for both systems. Table 4 lists the comparison of top

[11] https://github.com/dmlc/xgboost

| Features | Linguistic | | | | | | | | | SentiLexi | Domain-specific | | | | Word embedding | WCS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | unigram | bigram | trigram | rf_1 | rf_2 | rf_3 | Verb | NER | Word Cluster | SentiLexi | Number | Keyword+Number | Metadata | Punc | GoogleW2V | |
| Subtask 1 | √ | | | √ | | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | **0.7912** |
| Subtask 2 | | | | √ | √ | √ | √ | | √ | √ | √ | | | √ | √ | **0.7264** |

Table 3: Results of feature selection experiments for both subtasks on training datasets. rf_1, rf_2 and rf_3 stand for *rf_unigram*, *rf_bigram* and *rf_trigram* features respectively.

| Subtask 1 | | |
|---|---|---|
| Feature set | WCS | change (%) |
| Best features | 0.7912 | - |
| - rf_unigram | 0.7643 | -3.40 |
| - SentiLexi | 0.7664 | -3.10 |
| - metadata | 0.7841 | -0.90 |
| - number | 0.7869 | -0.54 |
| Subtask 2 | | |
| Feature set | WCS | change (%) |
| Best features | 0.7264 | - |
| - GoogleW2V | 0.6951 | -4.31 |
| - rf_unigram | 0.6964 | -4.13 |
| - SentiLexi | 0.7144 | -1.65 |
| - rf_bigram | 0.7245 | -0.27 |

Table 4: Ablation study: the comparison of top 4 most important features.

4 most important features.

From Table 3 and the ablation study results in Table 4, it is interesting to find that: (1) *rf_unigram* feature plays a key role in both subtasks and is more effective than *unigram* feature. The reason may be that *RF N-grams* features endow each word with a weight, which can capture how much the word contributes to the sentiment analysis of the sentence. Besides, the weight also contains some sentiment information. (2) *SentiLexi* features also make great contribution to both subtasks, which indicates that *SentiLexi* features are beneficial not only in traditional sentiment analysis tasks, but also in predicting the sentiment score of stocks in financial domain. (3) The *Number* features and *Keyword+Number* features are more effective in subtask 1 than subtask 2. The reason may be that there are plenty of numbers in the data of microblog domain but only a few numbers in news headline domain. (4) Although we only extract the *Metadata* features from the StockTwits, it perform better than most of other features, which indicates that the metadata is indeed significant. (5) The *GoogleW2V* feature is more effective in subtask 2 than subtask 1. The reason may be that the spans in microblog domain contain less words and many word vectors of the spans can not be obtained from the pre-trained *Google word2vec*. (6) The *bigram* feature and *trigram* feature are not beneficial in both subtasks. The possible reason lies in the large dimensions of these two features leading to sparse representation in two domains.

Overall, the system configurations for two subtasks are: using the optimum feature sets shown in Table 3 and the algorithms described in section 3.2.1 (i.e., ensemble with top 4 regression algorithm for subtask 1 and ensemble with top 2 regression algorithm for subtask 2) to build supervised regression models.

### 3.3 Results and Discussion on Test Data

| | Subtask 1 | Subtask 2 |
|---|---|---|
| Our system | 0.7779 | 0.7107 |
| Rank 1 | 0.7779 | 0.7452 |
| Rank 2 | 0.7600 | 0.7437 |
| Rank 3 | 0.7590 | 0.7327 |

Table 5: Performance of our systems and the top-ranked systems for two subtasks in terms of *WCS* on test datasets.

Using the system configurations described above, we train separate model for each subtask and evaluate them against the test set in SemEval-2017 Task 5.

Table 5 shows the results on test datasets. From Table 5, we find that: (1) Our system achieves lower performance on test data compared with the training data, the possible reason might be the different data distribution held between them. (2) Our results perform best among all submissions in subtask 1 and rank 5th in subtask 2, which proves the effectiveness of the method we proposed.

## 4 Conclusion and Future Work

In this paper, we extract four types of features, i.e., linguistic features, sentiment lexicon features, domain-specific features and word embedding features, and employ the ensemble regression model-

s to predict the sentiment score for two subtasks. The results on test and training data show the effectiveness of our method for this task.

For the future work, we would explore domain-specific sentiment lexicons and use the deep learning method (e.g., attention neural networks) to improve the performance. Due to the limitation of annotated data, we would like to first pre-train a neural network model on similar tasks (e.g., aspect-level sentiment analysis task), and then fine tune the neural network model on the current fine-grained sentiment analysis task to boost the performance.

## Acknowledgments

This research is supported by grants from Science and Technology Commission of Shanghai Municipality (14DZ2260800 and 15ZR1410700), Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (ZF1213) and NSFC (61402175).

## References

Keith Cortis, André Freitas, Tobias Dauert, Manuela Huerlimann, Manel Zarrouk, Siegfried Handschuh, and Brian Davis. 2017. Semeval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 519–535. http://www.aclweb.org/anthology/S17-2089.

Brian Davis, Keith Cortis, Laurentiu Vasiliu, Adamantios Koumpis, Ross McDermott, and Siegfried Handschuh. 2016. Social sentiment indices powered by x-scores. *ALLDATA 2016* page 21.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research* 9(Aug):1871–1874.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* pages 1189–1232.

Mengxiao Jiang, Zhihua Zhang, and Man Lan. 2016. Ecnu at semeval-2016 task 5: Extracting effective features from relevant fragments in sentence for aspect-based sentiment analysis in reviews. *Proceedings of SemEval* pages 361–366.

Man Lan, Chew Lim Tan, Jian Su, and Yue Lu. 2009. Supervised and traditional term weighting methods for automatic text categorization. *IEEE transactions on pattern analysis and machine intelligence* 31(4):721–735.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

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

Zhihua Zhang and Man Lan. 2016. Ecnu at semeval-2016 task 6: Relevant or not? supportive or not? a two-step learning system for automatic detecting stance in tweets. *Proceedings of SemEval* pages 451–457.

Tian Tian Zhu, Fang Xi Zhang, and Man Lan. 2013. Ecnucs: A surface information based system description of sentiment analysis in twitter in the semeval-2013 (task 2). *Atlanta, Georgia, USA* page 408.