Semester Project Report
Stock Prediction with Daily News

COMP3133 Chinese Language Computing
Department of Computing
The Hong Kong Polytechnic University

GU Jinshan, 14110914D
MA Mingyu Derek, 14110562D
MA Zhenyuan, 14111439D
ZHOU Huakang, 15050698D

April 22, 2018

Contents

1 Introduction

Automatic trading with the ability of quick response to the environment is a huge need for the financial industry, the rapid development of Artificial Intelligence and Machine Learning make this challenging topic realistic in recent a few years. In this project, we target at stock trend prediction task, and try to deal with this challenging problem using daily news posted on Reddit from a perspective of Natural Language Processing. We use traditional language features engineering approaches as well as deep learning method to increase the prediction accuracy. A automatic trading bot is also developed which connect to the prediction and classification model and utilize the carefully designed algorithm to make buy-sell decisions.

For the core classification and prediction module, we utilize several well-known classification models such as Naive Bayes, SVM, Decision Tree and so on to deal with the language features extracted using `sklearn` package. We also implemented and evaluated a Hierarchical Long Short-Term Memory Network method with pre-trained word vectors.

2 Existing Similar Systems and Applications

By previous researches, it suggests that there does exist relationship between stock prices and news articles. Thus there are a lot of similar systems predicting stocks using real-time news to seek the possible methodologies to gain money. We here select two as examples and briefly introduce them.

2.1 Stock Prediction by CUHK

In 2003, Gabriel hi Cheong Fung, Jeffrey Xu Yu and Wai Lam in The Chinese University of Hong Kong propose a stock prediction system based on Eficient Market Hypothesis Fung et al. (2003). This system could support multiple time series modeling and analyze textual documents automatically.

They first detect the trend of stock on time series instead of concentrating on the actual prices fluctuations. These information has been split into small segments, and after dealing with the oversegmentation, the adjacent segments are merged. Then by performing the feature extraction and applying the tf-idf scheme, these data are ready for training.

There are also something to do with stocks themselves. This group finds out the similar patterns among stocks and suggests that some stocks would affect the others. By this discovery, they select some stocks as representatives to investigate.

The training model they use is SVM (support vector machine) and use 33 stocks from 2002-10-1 to 2003-4-30 for experiment. The system can give a prediction with the cumulative profit higher than the cumulative loss and make the system both theoretically and practically possible to use.

2.2 Arizona Financial Text System

Robert P. Schumaker a and Hsinchun Chen Schumaker and Chen (2009) propose a quantitative stock prediction system based on financial news. They just use a synthesis of linguistic and financial and statistical techniques to generate a system named Arizona Financial Text System (AZFinText).

The procedures are similar comparing with the system mentioned in the last subsection. However, there are some differences in the detailed parts. The data source is from Yahoo! Finance they just extract proper nouns for later usage. They delete some nouns which appear twice or even less in the data. They also use some article term representing the stock price and the daily buy/sell recommendations from a variety of trading experts.

By using the SVR (support vector regression), they makes a prediction of what the price should be 20 min into the future for each news article. For experiment, they select 23 trading days between 2005-10-26 and 2005-11-28, and have predicted a 2% higher return than the best performing quant fund.

3 Language Features Approaches

3.1 Process Flow and Approaches

3.1.1 Data Import and Preprocessing

In this project, we use **pandas** to load the .csv data file. By using **pandas**, it is easy to split large data set into small groups such as "date", "label" and "news" for convenience. Date will be divided into two groups by before or after date "2015-01-01" as training or testing data sets.

As we notice that there are some abnormal news with character 'b' at the beginning and following with single or double quotation marks (i.e., ' or "). we use regular expressions tools to preprocessing the news group into normal sentences without punctuations. However, we remain hyphens (i.e. -) as information in some words may be lost if we remove hyphens.

3.1.2 Features Extraction

The data sets available to us are in the form of the 25 news title strings for each day with its label and 1990 days in total. Since, in this classification problem, the features are the words or terms in the news titles, we need a bag of terms as the feature data for each day, instead of 25 strings. Therefore, 25 strings are concatenated together to obtain one longer string, which is then transformed to a bag of terms.

Standord **corenlp** package is used to process the terms so as to adjust the importance of all features in the model. Despite that there are many useful annotators provided in corenlp, most of them are useful for syntactical and semantic analysis. As a result, only the annotator **lemma** is used to change the words to its lemma (root form), if it is possible.

After the preprocessing of origin data sets, we used **CountVectorizer** to transform to strings to a vectors of features, which are all terms in the string. To be more precise, all distinct 2-words terms are put into the vector by specifying the parameter ngram_range=(2,2).

Feature selection is necessary to remove irrelevant features so as to hasten the model training as well as to simplify the model. Two thousands most important features are selected using the **SelectKBest** class and **Chi Square** test function, and generate a mask (filter) for features. Later, in the prediction, only features which go through the best feature filter are considered.

3.1.3 Classification

In this project, the data sets are splited into two groups, training (before the year 2015) and testing (2015-01-01 or later). Different models are used in the experiment to find the most suitable model to this classification problem.

3.1.4 Evaluation and Visualization

From the prediction on the testing data sets, the model can output a vector of the prediction of the class for each sample. With the true label of testing

data, we can obtain statistics information about the performance, including precision, recall, F-score and support for each class (1 and 0), as well as a confusion matrix of the classification. A heatmap (Figure 1) can be generated from the confusion matrix to demonstrate the distribution of prediction.

Confusion Matrix		Predicted	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 1: Confusion Matrix

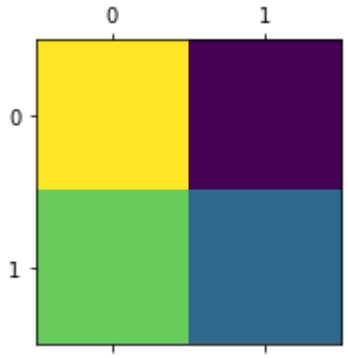

Figure 1: An example of heatmap

The models can also output all the probabilities of each sample being in each class. With the true labels vector and the probability vector, a ROC curve can be plotted and an AUC value is calculated. A ROC curve (Figure 2) is created by plotting true positive rate (TPR) against false positive rate (FPR). There is a significant benefit of using ROC curve instead of a simple confusion matrix, since the curve can show performances at all possible thresholds. It also gives a help to find the optimal threshold for the classification problem. AUC standing for area under the curve, which is the percentage of the area the curve in the graph, is another representation as ROC curve.

$$TPR = Sensitivity = \frac{TP}{TP + FN}$$

$$FPR = 1 - Specificity = 1 - \frac{TN}{TN + FP}$$

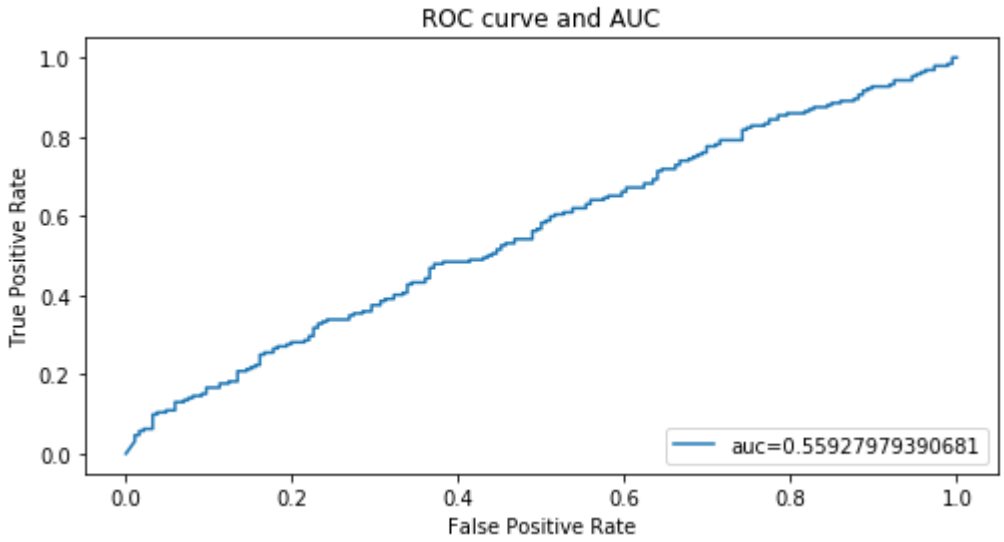

Figure 2: An example of ROC curve graph

3.2 Classification Models

We use a large amounts of classification models to value the performance of the system. There is a Python package named "sklearn" which contains simple and efficient tools for data mining and data analysis. We choose some proper ones from the package and list below:

1. Multinomial Naive Bayes: It is suitable for classification with discrete features such as word counts. Also, as fractional counts such as tf-idf may work, this model gives really higher precision.

2. Bernoulli Naive Bayes: Like MultinomialNB, it is suitable for discrete data. The difference is that it is designed for binary/boolean features. Therefore, it cannot be applied directly to the system.

3. Random Forest: A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

4. Gradient Boosting Machines: It builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage the regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function.

5. Ada Boost Classifier: An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

6. Gaussian Naive Bayes: It can perform online updates to model parameters via partial_fit method.

7. Linear Support Vector Classification: It has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

8. Nu-Support Vector Classification: Similar to SVC but uses a parameter to control the number of support vectors.

9. Neural Network Multi-layer Perceptron classifier LBFGS: optimizer in the family of quasi-Newton methods:

10. Neural Network Multi-layer Perceptron classifier SGD: stochastic gradient descent.

11. Neural Network Multi-layer Perceptron classifier ADAM: a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba.

12. Decision Tree: It is a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Some of the models work really well as the data set may be more suitable. We list the system precision with and without Stanford Corenlp packagehere:

Model	Precision with Corenlp	Precision without Corenlp
Multinomial Naive Bayes	0.47	0.53
Bernoulli Naive Bayes	0.45	0.53
Random Forest	0.50	0.53
Gradient Boosting Machines	0.48	0.46
Ada Boost Classifier	0.55	0.47
Gaussian Naive Bayes	0.49	0.53
Linear SVC	0.49	0.53
Nu-SVC	0.48	0.50
MLPClassifier lbfgs	0.48	0.52
MLPClassifier sgd	0.56	0.54
MLPClassifier adam	0.44	0.52
Decision Tree	0.52	0.51

Table 2: Precision of 12 models

By comparing the precision, we find that the model Neural Network Multilayer Perceptron classifier SGD performs best no matter we use Stanford Corenlp package or not. Moreover, it is hard to say whether there are benefits when using the feature extraction by Stanford Corenlp package as the precisions do not have a natural trend.

4 Deep Learning Method: Hierarchical Long Short-Term Memory Networks (HLSTM)

In this section, a deep learning method for stock price prediction is introduced. We use Hierarchical Long-Short-Term Memory Networks to construct the network and the detialed structure will be introduced in the following.

The implementation of the deep learning model in this section using `Python` with `TensorFlow` can be found at GitHub: https://github.com/derekmma/stock-pred.

4.1 Model Design

In the traditional approaches, n-gram is used to extract language features. But much information in the document is lost because the sparse representation and loss of sequential information in traditional language modeling methods. Long Short-Term Memory (LSTM) is a good solution for this issue (Hochreiter and Schmidhuber, 1997). One hand, the neural network can

keep much more features then traditional feature engineering (Tang et al., 2015b), another hand, LSTM provides a start-to-end hidden state to link all the cells together so that the long-term memory performance is significant improved compared to other solutions (Tang et al., 2015a). A detailed internal structure of LSTM cell is shown in Figure 3 [1]. Since for our dataset,

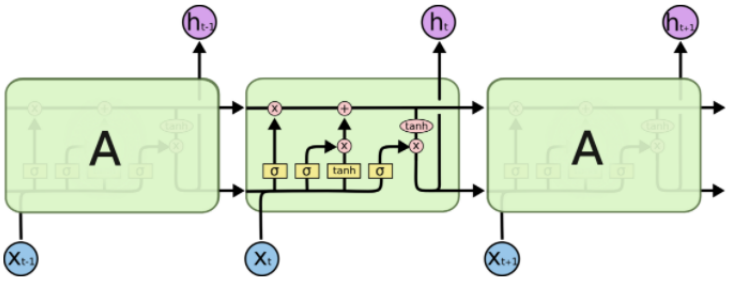

Figure 3: Internal structure of LSTM cell

25 sentences is quite long, if we link all 25 headlines together, the vectors is too long which may hard to get fully trained in the model. So a approach to deal with long sequential data is a huge need for this problem. We found hierarchal LSTM is widely used in the research community to do document embedding or sentiment analysis (Hochreiter and Schmidhuber, 1997). In this kind of solution, different level of data are represented and trained by different layers, and the evaluation results also shows this method is quite efficient. Inspired by the successful usage of hierarchal LSTM structure in the works of (Chen et al., 2016), we utilize the following HLSTM structure to predict the stock trend.

The idea of this method is that two LSTM layers will in charge of the conversion from word-level to sentence-level and sentence-level to document-level respectively. Firstly, each word can be embedded into a vector, then the output will be the final prediction through this network. Through multi-layer LSTM, features can have the chance to be fully trained.

Assume the embedding dimension is n, then each word is embedded into a vector with size $1 \times n$. Assume a sentence have num_w words inside, then the input matrix to the first LSTM layer is of size $num_w \times n$. From the hidden state of the first LSTM layer, a vector of this sentence with size $1 \times n$ can be obtained which is considered as the sentence representation. If we consider the all 25 news headlines of a day is a whole document, then the document

[1]The diagram is from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

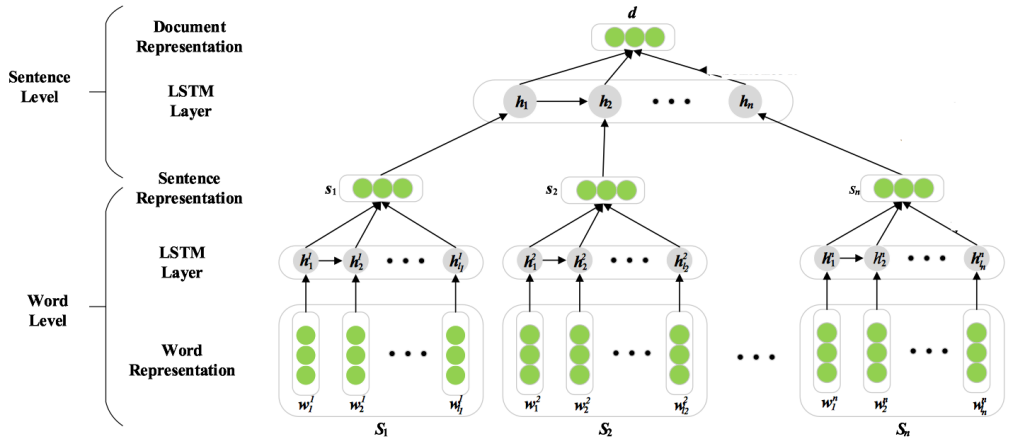

Figure 4: Model Design for HLSTM (Chen et al., 2016)

consists of 25 sentences which already been represented by the hidden state of the first LSTM layer as a $1 \times n$ vectors. Then the input of the second LSTM layer $25 \times n$ matrix. From the hidden state of the second LSTM layer, we can obtain the vector with size $1 \times n$ which is considered as the final representation of the document.

After we get the output of each LSTM layer, a pooling layer and drop out layer will be went through. Finally, a linear function is went through which contains trainable weight and bias. The final prediction can be obtained by a final $Softmax$ function. The final prediction mechanism is shown in Equation 1.

$$Output_{HLSTM} = Softmax(W * Output_{second\ LSTM\ layer} + b) \tag{1}$$

The loss function of the whole network is defined as the cross entropy between prediction results (0/1) and ground truth label. By minimizing the loss function with a Adam optimizer, the parameters in the network can be optimized.

4.2 Word Embedding

The first step of the model is to embed words into vectors. Simple Bag-of-Words (BoW) method can be utilized, but definitely lots of information will

be lost. Since BoW will not consider the sequence of words in the sentences. In this task, we utilize "GloVe", global vectors for word representation proposed by Pennington et al. (2014) at Stanford University. Pre-trained word vectors using Wikipedia, Twitter corpus can be directly downloaded [2]. Due to limited computational resources, we take the smallest set of GloVe as our word embedding solution.

The version of GloVe we used in this project is GloVe.6B. A Wikipedia 2014 and Gigaword 5 pre-trained word vectors are used in the following experiments. There are 400K vocabulary and we utilize the ones with the dimension size of 50.

4.3 Evaluation and Analysis

We run the HLSTM on a GPU and tested the model with same configuration for three times. The best accuracy achieved by HLSTM is 0.580729166666666. Some observations can be found from the experiments.

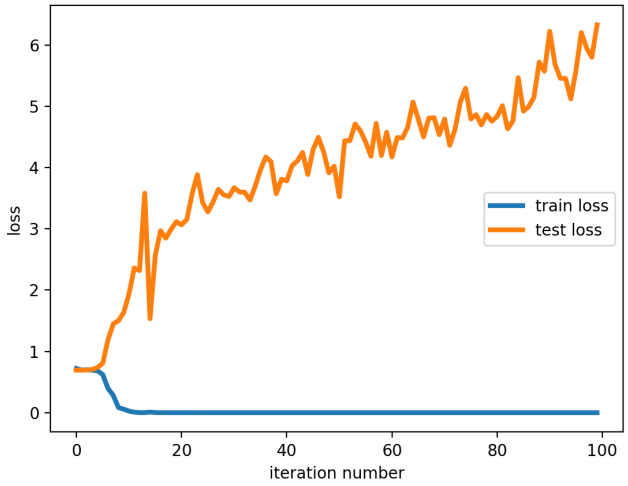

Figure 5: Training and testing dataset loss across the learning

Figure 5 shows the loss of training set and testing set through the 100 iterations. It clearly shows that after around 10 iterations, the training loss stays

[2]Word vectors in this project is downloaded at: https://nlp.stanford.edu/projects/glove/

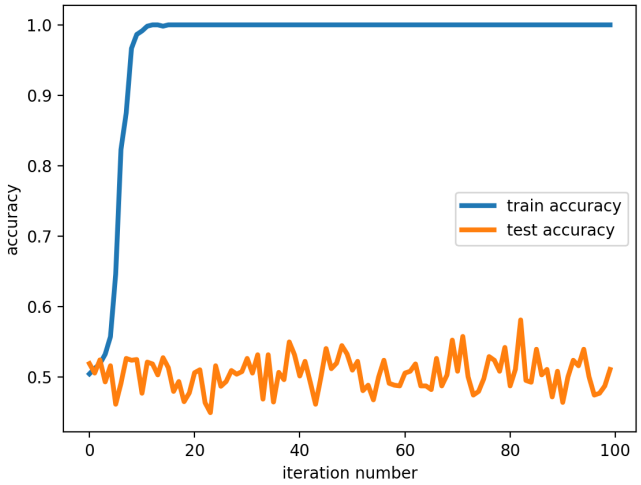

Figure 6: Training and testing dataset accuracy across the learning

at a very small value, while the testing loss increases continuously. Since the algorithm is optimized for the training loss, so the algorithm will not be optimized even the testing loss is still very large.

Figure 6 shows the accuracy of training set and testing set. We can find that training accuracy stays at 1.0 after around 20 iterations. This means all the prediction for training set is correct. While the testing set accuracy varies all the time.

The data in the training process clearly shows that the over-fitting problem is very serious. The model performs extremely great on training set and leave little space for the model to improve. The possible cause for this issue is that the amount of data is very limited but the amount of parameters inside the model is comparable very large.

Possible solution would be simplify the model and decrease the parameters amount, or add more data in the model from other data source. The limited amount of data is the constrain for the performance of hierarchical LSTM model.

5 Investment Robot Design

Based on the prediction result, we design an investment robot to simulate the investment process in stock market. For simplicity purpose, we assume the stock price only increases or decreases for 1% on each day; And we have $10000 as initial money. We simulate the investment according to the predicted result generated by the best classification model.

We use two algorithms in the investment robot. For the first algorithm, we only care about whether the predicted result is "increase" or "decrease". And we assume the predicted result is always correct. That means the robot will spend all money to buy the stock if the model predicts that the stock price will increase; otherwise the robot will sell all stocks. The second algorithm is more cautious. It utilizes the probabilities generated by the prediction model. Besides the "0" or "1" prediction, the classification model can also generate a real number p between 0 and 1, represents the model is sure about the predicted result with p probability. In the second algorithm, if the predicted result is "increase", the robot will use p percents of the total assets (including the cash and the money in the stock market) to buy stock, and keep $1 - p$ percents of the total assets as cash; If the predicted result is "decrease", the robot will put $1 - p$ percents assets into stock market and keep the remaining as cash. The intuition of this algorithm is that if the robot is not certain about the prediction result, it will be more cautious.

The user is asked to input start date and end date of the investment. And we use the data before the start date to train the classification model. In the experiment, we test the robot using 2 years as the investment period, and evaluate the result by the final money the robot has after investment. The simulation results are shown in Table 3.

Investment period	Final money using algorithm 1	Final money using algorithm 2
2010 - 2012	18811	15668
2012 - 2014	10029	13275
2014 - 2016	13521	11724

Table 3: Investment robot experiment results

We can found that the two algorithms give different performances in different periods. But both of them can earn money in these test cases. This

shows that our classification model works quite will.

6 Role and the Contribution of Each Group Member

GU Jinshan: Trading bot design and development, experiments of different classification models
MA Mingyu: Deep learning methods literature review, deep learning model design, deep learning model implementation and evaluation
MA Zhenyuan: Data importing and cleaning, add different models for classification in language features methods
ZHOU Huakang: Feature extraction, performance evaluation and visualization.

7 Conclusion

For using the traditional language features methods, we can find that among all the classification models, Neural Network Multi-layer Perception classifier SGD performs the best, and it achieves the accuracy of 0.56 with the data cleaning by Stanford Corenlp suit. Also there is no clear clue to show that the data cleaning process done by the Stanford Corenlp package helps.

For deep learning method, we proposed a Hierarchical Long Short Term Memory Network model for stock trend prediction. In that model, two LSTM layers convert each individual word embeddings obtained from GloVe to document level representations. Finally the binary prediction can be obtained by a linear and $Softmax$ function. The HLSTM model achieve the accuracy of 0.581. Through observation of the experiments, we can find that due to limited data amount, over-fitting problem is very serious for HLSTM working on this small dataset. Possible future directions include simplifying the model, adding more related data and utilization of reinforcement learning for text classification task.

An automatic trading bot is developed. It can achieve automatic trading after inputting the start and end date for investment.

References

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. Neural sentiment classification with user and product attention. In *EMNLP*, pages 1650–1659, 2016.

G. Pui Cheong Fung, J. Xu Yu, and Wai Lam. Stock prediction: Integrating text mining approach using real-time news. In *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.*, pages 395–402, March 2003. doi: 10.1109/CIFER.2003.1196287.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Robert P. Schumaker and Hsinchun Chen. A quantitative stock prediction system based on financial news. *Information Processing & Management*, 45 (5):571 – 583, 2009. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2009.05.001. URL http://www.sciencedirect.com/science/article/pii/S0306457309000478.

Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015a.

Duyu Tang, Bing Qin, and Ting Liu. Learning semantic representations of users and products for document level sentiment classification. In *ACL (1)*, pages 1014–1023, 2015b.