Aiding Sentiment Evaluation with Social Network

Pengfei Gao, Fan Yang, Hao Yin *
Institute for Computational and Mathematical Engineering
Stanford University
Stanford, CA 94305
{pfgao, fanfyang, yinh}@stanford.edu

Abstract

One challenge for sentiment analysis as well as many other natural language processing task is the language variation. In this project, based on the intuition of language homophily, we aim to solve this problem by combing social network information. We propose a new model that measure the interaction between author embedding and sentence words' embeddings. This model is based on the current state-of-the-art methodology that does not consider social network information. The authors' embeddings are pretrained with three different yet most popular network node embedding methods, and their performances are compared.

1 Introduction

Sentiment analysis is one of the important tasks in natural language processing community[8] which helps people navigate the huge amount of user-generated content available online. Machine learning systems that make decision on the attitude of viewpoints to be positive, neutral or negative that enable people to understand the enormous body of opinions on the internet, ranging from product reviews to political positions.

One of the biggest challenges in sentiment analysis, as well as in almost all fields for natural language processing research, is the language variation. Words can mean different things to different people, and different people express their feeling and idea in a different way. However, such variation is believed to be tractable from social factors [1]. For example, people of the same ago may speak in a similar way [12] which might be influenced from their youth education, and people from the same community use the same language which is known as jargon. Therefore, social network information provides additional information to solve the problem in language variable thus improve the general prediction performance in natural language processing tasks.

Online social networks provide promising platforms to study the language variations. Most online websites have social network behind it, and user-generated content often appears in the context of social media. Therefore nowadays user-relationship information is now more easily obtainable. For example, huge amount of tweets from Twitter express people's opinions on different subjects. Each tweet is associated with a user and users formed social network structure through the mechanisms of "follower". When a user forms a link in the network such as Twitter, they tend to have a personal relationship then the principle in language called "homophily" suggests that users who are connected via some social relationship may also share similar opinions or linguistic variation (each community may have their own "jargon" in expressing ideas and sentiments). Figure 1 from [1] gives an example of how users from different communities may understand the word "sick" differently.

Nowadays, models that combine social network information with machine learning classification task are proposed in many literature. However, these models rarely utilize the state-of-art deep

*Each member contributes equally, and names are put in alphabetic order.

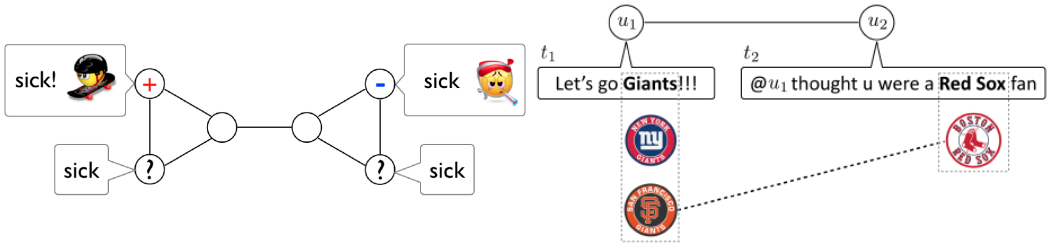


Figure 1: Words such as sick can express opposite sentiment polarities depending on the author; Leveraging social relations for entity disambiguation.

learning methods like convolutional neural network or network node embeddings which are common in social network and NLP communities. Previous research either use traditional machine learning methods to incorporate social network structure information as in [7] or they separate the textual and user information to build separate deep learning model as in [8]. None of the previous works directly model the interaction between author information, especially the social network information, and the sentence context. In this paper, we are going to explore different methods that utilize jointly social network information and textual information in sentiment analysis with one joint deep learning model that takes both social network information and textual information as input to classify the sentiment of sentences.

Our paper is structure as follows: section 2 will introduce related work to our task and how we relate them; section 3 will define the problem formally and introduce our dataset; section 4 will introduce our model and intuition; section 5 will demonstrate our numerical result; section 6 will discuss our result and draw our conclusion.

2 Related Work

Sentiment Analysis

The current state-of-the-art method is convolutional neural networks(CNN)[9] which takes word embeddings from sentences as inputs and output a softmax classification to identify sentence sentiment. The typical structure of such CNN is some convolutional layer on top of original sentence word embeddings, then a max pooling layer on top of the convolutional layer to extract some extreme information. Finally, a dense layer with fully connected network is added to transform features from CNN to a softmax classifier. A simple CNN model with one convolutional layer of two-width window plus one max pooling layer with single channel can achieve amazing result[1]. Different initializations methods[11] could also be adopted to improve prediction accuracy, but they all share the similar structure as described above.

Network Node Embeddings

The emergence of various network node embedding methods make it possible to represent each node with a meaningful vector representation. The aim is to assign vectors such that connected nodes have similar vector while socially distant pairs of nodes have different vectors. The idea in achieving this vector assignment originated from the skip-gram word2vec method, which aims to maximize the likelihood of occurrence of context words given the center of each sentence window. In various node embedding methods, one aims to maximize the likelihood of node 1-hop neighborhood based on the center node. With different neighborhood sampling methods and objective function, the three current most popular efficient algorithm that easily applies to network of thousands of nodes are the DeepWalk [2], LINE [3], and node2vec [4]. These three model have proven to obtain good performance in many downstream prediction tasks such as multi-label classifications. In this paper, we compare the performance of these three methods in solving the language variation problem in sentiment analysis.

Aiding Classification with Social Network

The intuition behind combing social network with classification is that users connected are more likely to hold similar opinions and use language similarly.

Tan et al. (2011) [5] is the first paper to show social relationship information can be exploited to improve sentiment analysis. They have shown numerically that incorporating social-network information can indeed lead to statistically significant sentiment-classification improvements over the performance of a SVM baseline model that only has access to textual features. Yang and Eisenstein(2016)[1] is a more recent version for combining social network information and sentiment analysis. They study task for classifying sentiment to be positive, neutral and negative for each tweets given text and user ID information. Their model consider the author information and sentence information separately: each node (author) in the network is assigned an embedding vector using the LINE algorithm[2], and then is (softly) assigned each cluster on the network based on the embedding. Each cluster has its own model, which is a CNN model combined with max pool layer. Detail model specs are described in section 4 as a comparison to our model.

On the other hand, Yang and Chang(2016)[6] study another problems called entity linking, which is the task of identifying mentions of entities in text, and linking them to entries in a knowledge base. They achieve the-state-of-art result with a tree-based model in Twitter data. To further improve the performance, Yang et al.(2016)[7] propose to incorporate social network information in the same problem. Intuitively, socially linked individuals share interests, and are therefore likely to mention the same sorts of entities. They build a bilinear model based on the previous the-state-of-art tree model[6] that consider interactions of users and entities. This new model incorporating social network information has a F1 improvements of 1%-5% on benchmark datasets.

3 Data Description

Yi and Eisenstein have provided the data used in [1]. The data consists of a collection of tweets as well as some network information on Twitter.

3.1 Corpus

The corpus contains a collection of samples (tweets). For each data sample, we have one tweet ID, one user ID, a sentiment label (positive, neutral, negative), and the tweet content itself. For example, the following are two examples of our data samples.

```
261140278944088066 17572408 negative @USER may i have an ...
237571817550786563 727519172 neutral @USER i told you shane ...
```

The size of the corpus is the following:

Dataset	# Positive	# Negative	# Neutral	# Tweet
Train 2013	3,230	1,265	4,109	8,604
Dev 2013	477	273	614	1,364
Test 2013	1,572	601	1,640	3,813
Test 2014	982	202	669	1,853
Test 2015	1,038	365	987	2,390

Figure 2: dataset description

3.2 Network

We have contacted Yi, one of the author of [1], to obtain the three social networks he used in the paper, i.e., FOLLOWER, MENTION and RETWEET network. The construction of each network is seen in [1]. Each network is an edge list of form:

```
73225701 8161232
95679562 87818409
...
```

On each line, the two numbers are user IDs, indicating that there is an edge (direct connection) between the two nodes. The first and second are two user IDs which are consistent with the above twitter data. These two user IDs indicate these two users are connected in the network.

Three social network structures were constructed in [1]. As described in the original paper:

> *We construct three author social networks based on the follow, mention, and retweet relations between the 7,438 authors in the training dataset, which we refer as FOLLOWER, MENTION and RETWEET. Specifically, we use the Twitter API to crawl the friends of the SemEval users (individuals that they follow) and the most recent 3,200 tweets in their timelines. The mention and retweet links are then extracted from the tweet text and metadata. We treat all social networks as undirected graphs, where two users are socially connected if there exists at least one social relation between them.*

4 Methodology

Our task is given tweet content "i told you shane would get his ..." and user ID "72751", we need to classify the sentiment of this sentence to be positive, neutral or negative. Our model will start from the state-of-the-art convolutional neural network model. Upon CNN baseline, we modify it with a author embedding activation layer.

CNN Baseline

Sentence word embeddings are concatenated together as inputs, $(\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_n})$, where the maximum sentence length is usually less than n words due to character constraints of tweets. If a sentence is less than n words, we will use zero word vectors to fill it. Therefore, our input data are consistent length n. Next, a convolutional layer is performed on the sentence word embeddings with a width-two window to d-channel filters of size $n-1$. Then a max pooling layer pick the largest value among the $n-1$ layers to form a d-channel features. Finally, this length d features are input into a softmax layer to classify three-class label. To express the above model structures in mathematical equation:

1. The convolutional layer:

$$\mathbf{c_i} = \tanh(W_L \mathbf{h_i} + W_R \mathbf{h_{i+1}} + \mathbf{b}), i = 1, 2, \ldots, n-1 \tag{1}$$

   where $\mathbf{h_i} \in \mathbb{R}^{D \times 1}$ are word embedding, $\mathbf{c_i} \in \mathbb{R}^{d \times 1}$ are padding i in the convolutional layer, W_L and W_R are convolutional layer kernel matrices, $\mathbf{b}$ is the convolutional layer bias vector.

2. The max pooling layer:

$$\mathbf{s} = \max_{i=1,2,\ldots,n-1} \mathbf{c_i} \tag{2}$$

   where $\mathbf{s} \in \mathbb{R}^{d \times 1}$ are features learned from CNN.

3. The softmax layer:

$$\mathbb{P}(Y = c|s) = \frac{\exp(W_c \mathbf{s} + \mathbf{b_c})}{\sum_{c'} \exp(W_{c'} \mathbf{s} + \mathbf{b'_c})} \tag{3}$$

Author Activated CNN

In our dataset, each tweet also includes a user ID. We use algorithm of LINE, DeepWalk and Node2Vec to learn a network node embedding for each user ID to form author embeddings $\mathbf{a}$ for each sentence $(\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_n})$. Compared to baseline CNN, we use one hidden layer to learn an activated vector $\mathbf{z}$ for each user which element-wise multiplies convolutional layer. The intuition

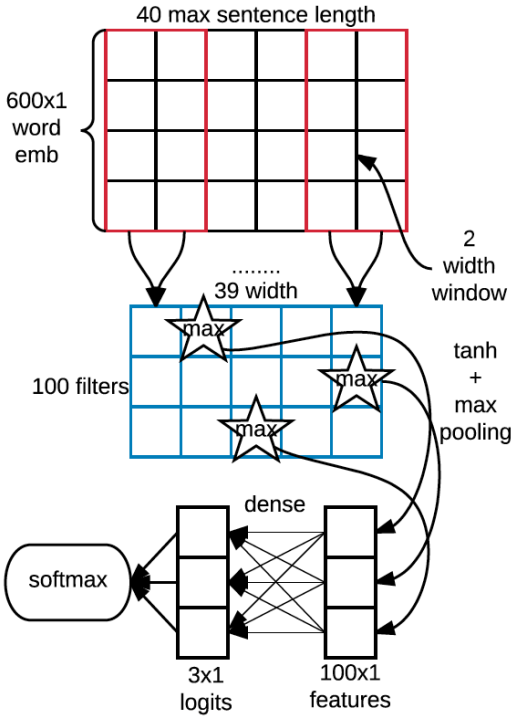


Figure 3: baseline CNN network strucutre

behind this activated vector is that for each convolutional channel, different user may display different behaviors. This is in fact equivalent with bilinear form with low rank approximation. We can exploit the interaction between word embeddings and author embeddings. To express the above model structures in mathematical equation:

1. The convolutional layer:
$$\mathbf{c_i} = \tanh(W_L \mathbf{h_i} + W_R \mathbf{h_{i+1}} + \mathbf{b}), i = 1, 2, \ldots, n-1 \tag{4}$$
where $\mathbf{h_i} \in \mathbb{R}^{D \times 1}$ are word embedding, $\mathbf{c_i} \in \mathbb{R}^{d \times 1}$ are padding i in the convolutional layer, W_L and W_R are convolutional layer kernel matrices, $\mathbf{b}$ is the convolutional layer bias vector.
2. The user activated layer:
$$\mathbf{z} = \tanh(W_a \mathbf{a} + \mathbf{b_a}) \tag{5}$$
where $\mathbf{z} \in \mathbb{R}^{d \times 1}$ are activation features learned from author embeddings $\mathbf{a} \in \mathbb{R}^{A \times 1}$.
3. The max pooling layer with user activation:
$$\mathbf{s} = \max_{i=1,2,\ldots,n-1} \mathbf{z} \odot \mathbf{c_i} \tag{6}$$
where $\mathbf{s} \in \mathbb{R}^{d \times 1}$ are features learned from CNN.
4. The softmax layer:
$$\mathbb{P}(Y = c|s) = \frac{\exp(W_c \mathbf{s} + \mathbf{b_c})}{\sum_{c'} \exp(W_{c'} \mathbf{s} + \mathbf{b'_c})} \tag{7}$$

Note that the difference between node activated CNN and baseline CNN is that activated CNN has new parameters W_a and $\mathbf{b_a}$. In fact, originally we would like to build bilinear form for each word vector $\mathbf{h_i} W_{ha} \mathbf{a}$ to form some interaction features for each word, hoping the interaction features could help identify different behaviors for each user community. But we find out our data sample is too small to learn this bilinear matrices W, therefore we have to use some low rank approximation of W which can be equivalent to element-wise activation as described above.

	CNN	SA [1]	DeepWalk	LINE	node2vec
Dev2013	68.85	69.52	67.71	69.51	68.58
Test2013	69.53	69.98	67.58	69.67	68.58
Test2014	72.41	72.70	71.46	71.44	71.46
Test2015	64.40	65.28	64.71	64.57	64.25
Avg test sets	68.78	69.32	67.92	68.56	68.10

Table 1: Prediction performance on each Dev and Test Sets.

5 Experiments

We implement our model in Section 4 with TensorFlow, and apply our code on the dataset described in Section 3. To be specific, we train and tune our model on the 2013 Train and Dev dataset, and evaluate our model on the Test 2013–2015 dataset. The focus of this section is to compare our author activated CNN model performance with the baseline CNN model.

5.1 Experiment Setting

We employ the pretrained word embeddings by Austudillo et al. [13]. These embeddings are obtained via training on a corpus of 52 million tweets using the structure skip-gram model [14], and have been shown to perform very well on sentiment analysis tasks [1]. We also use the same evaluation metric as the SemEval challenge.

We pretrain the word embedding using several existing network node embedding methods including DeepWalk [2], LINE [3], and node2vec [4]. These methods are applied to all the FOLLOWER+, MENTION+, and RETWEET+ networks. In addition, to provide a baseline for different node embedding methods, we use purely random node vector.

Regarding baseline models, we adopt the state-of-the-art convolutional neural network model methodology for sentiment analysis. This model is the basis model of our Author Activation CNN Model and has been described in Section 4. As a second baseline model, we adopt the Social Attention model [1] which models the soft-assignment of users to communities.

Parameter tuning For both the baseline CNN model and author activated CNN model, we tune all the hyper parameters on the SemEval 2013 Dev dataset, including

- number of bigram filters for the CNN models, from {16, 32, 50, 100};
- dropout rate, from {0.1, 0.2, 0.4};
- L_2-penalty coefficient, from {1e-6, 1e-4, 1e-2};
- author embedding dimension, from {16, 32, 50, 100}.

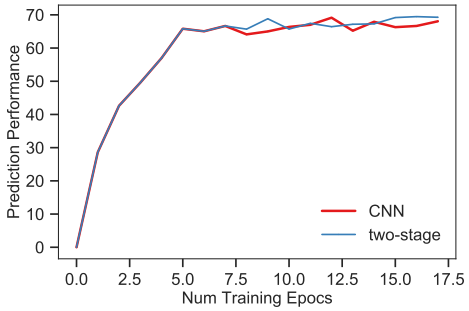
Moreover, we precomputed the author embeddings on all the FOLLOWER+, MENTION+, and RETWEET+ networks with three node embedding methods. Using different networks as well as embedding algorithms can all be considered as hyper parameters. We also adopted early stopping method based on Dev Performance to prevent our model from overfitting.

For the Social Attention model, we use the best hyper parameter set provided by the author [1].

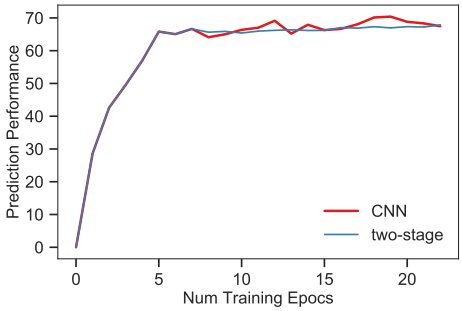
5.2 Results

Table 1 summarizes the empirical result of our experiments. Both baseline CNN model and our author activated CNN model are tuned for all parameters The best hyper parameter sets for baseline CNN model is 100 bigram filters, 0.4 dropout rate, and 1e-2 for L_2 penalty coefficient. The best

[1] Social Attention model [1]. Note that even after contacting the authors and obtaining their code, we still fail to retrieve their declared result in the paper. Here we put the implemented result using their provided code.



Updating both convolution layer weights and author layer weights



Fix convolution layer and only train author layer weights

Figure 4: Performance of two-stage training method.

hyper parameter sets for the author activated CNN model is 100 bigram filters, 100 author embedding dimension, and from RETWEET+ network. The best dropout rate and L_2 penalty coefficient depends on which node embedding method we use. From the table, we see that the improvement after adding the node embedding is insignificant. The author-activated CNN model sometimes do even worse in terms of test sets prediction. Different node embedding scheme does not seems to have big impact on the downstream prediction task.

5.3 A two-stage training method

We believe the reason for unsatisfying result is that author informations are considered too early in our training process. We combine author information to solve the problem of language variation. However, immediately after we randomly initialize the weights in baseline CNN model, language variation can not be seen, i.e., we are not more likely to make wrong prediction to twitter between a pair of friends. At this time, most gains are achieve by training the sentence interpretation part (encoded by baseline CNN model) rather than solving language variation. Therefore, the update on author embedding weights are almost purely random thus will harm our model in the long run. The language variation is better heard after the sentence interpretation part is mature. In the training process, this is the time when the baseline CNN model start to overfit.

Therefore, we propose a two-stage training method. In the first stage, we fix the weights related with author embedding, and only train the weights existing in the baseline CNN model. After several epochs of training, the improvement of development set prediction performance starts to fluctuate (See Figure 4), we interpret this an a sign of baseline CNN model overfitting, and it is only at this stage, the second stage, that we start to train the layer related with author embedding.

Table 4 demonstrates the performance of two-stage training method. We consider two variations. First, in the second stage we update the weight matrices in both sentence convolution layer and author embedding processing layer. Second, in the second stage, we fix the sentence convolution layer and only update the author embedding layer. Unfortunately, in both variation, the improvement in prediction performance from adding author embedding information is not significant. In the first variation (left plot of Figure 4) the prediction performance still fluctuate just like the CNN training method; in the second variation, there seems to be no improvement.

6 Discussion and Conclusion

Our experiment results are not cheerful, and we believe it is due to the following two reasons. First, our sample size is too small, which makes the training suffer from overfitting. Even if we have employed different overfitting prevention techniques including L_2 penalty on weights, dropout, and early stop, considering that we only have 8,000 training tweet samples it is not likely to build insightful models. We believe a larger training set would help solving this issue of overfitting.

Second, the experiment results show that the twitter networks are not informative in terms of language variation. One follow another person does not mean that these two people are friends and should have similar language usage preference. We believe a better network would be Facebook friendship network.

The experiment by Yang and Esteintin(2016)[1] is not persuasive either. We believe their model improvement is due to ensemble effect. In fact, in our running of their code, their baseline models outperform their social attention model.

Despite the unsatisfying result, we still believe social network information may help solving the issue of language variation and thus improve the general performance of sentiment analysis and other natural language tasks. Even though this user-activation method does not help the CNN baseline model, other ways of introduction may help, and may also help the prediction of other NLP models such as RNN and LSTM.

Reference

[1] Yang, Yi, and Jacob Eisenstein. "Overcoming Language Variation in Sentiment Analysis with Social Attention." arXiv preprint arXiv:1511.06052 (2016).

[2] Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.

[3] Tang, Jian, et al. "Line: Large-scale information network embedding." Proceedings of the 24th International Conference on World Wide Web. ACM, 2015.

[4] Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016.

[5] Tan, Chenhao, et al. "User-level sentiment analysis incorporating social networks." Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011.

[6] Yang, Yi, and Ming-Wei Chang. "S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking." arXiv preprint arXiv:1609.08075 (2016).

[7] Yang, Yi, Ming-Wei Chang, and Jacob Eisenstein. "Toward socially-infused information extraction: Embedding authors, mentions, and entities." arXiv preprint arXiv:1609.08084 (2016).

[8] Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." Foundations and Trends in Information Retrieval 2.12 (2008): 1-135.

[9] Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).

[10] Dos Santos, Ccero Nogueira, and Maira Gatti. "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts." COLING. 2014.

[11] Severyn, Aliaksei, and Alessandro Moschitti. "Twitter sentiment analysis with deep convolutional neural networks." Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2015.

[12] Rosenthal, Sara, and Kathleen McKeown. "Age prediction in blogs: A study of style, content, and online behavior in pre-and post-social media generations." Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011.

[13] Astudillo, Ramn Fernandez, et al. "Learning Word Representations from Scarce and Noisy Data with Embedding Subspaces." ACL (1). 2015.

[14] Ling, Wang, et al. "Two/Too Simple Adaptations of Word2Vec for Syntax Problems." HLT-NAACL. 2015.
APA