Joint Embedding of Words and Labels for Sentiment Classification

Yingwei Sheng
University of Tsukuba
Tsukuba, Japan
fireboat42@gmail.com

INUI Takashi
University of Tsukuba
Tsukuba, Japan
inui@cs.tsukuba.ac.jp

Abstract—Due to the fast growth of social networks, sentiment analysis on the web has been a popular research topic. Recently, word embedding-based sentiment analysis methods have reached outstanding performance compared to traditional methods. However, word embeddings always ignore information from dataset’s labels. Inspired by LEAM model proposed by Wang [1], we propose a method that jointly learns information of words and sentiment labels, which can improve the performance of the label embedding model. We defined a set of sentiment lexicons and used it to represent sentiment labels in the proposed method. We finally conducted experiments on Yelp dataset, which reached 64.99% accuracy.

Index Terms—sentiment analysis, label embedding

I. INTRODUCTION

Sentiment analysis is an important area in natural language processing. The goal of sentiment analysis is to annotate a given text with one or multiple labels that represents its sentiment tendency. Traditional methods are usually rule-based, like Vader [2]. With advances in machine learning, word embeddings have been widely applied in NLP tasks. For example, Zhang [3] proposed a method for sentiment classification based on Word2vec [4] and SVM [5], and achieved accurate classification performance. Those methods capture semantic information from words, and can represent words or sentences in vector formulations. To furthermore capture sequence information in texts, Wang [6] proposed a method using disconnected recurrent neural network, which achieved state-of-art performance on several benchmark datasets for text classification.

There are many different types of word embedding methods. A typical work is Word2vec introduced by mikolov [4]. Word2vec can capture similarities between words, thus words with similar meaning can have similar vector representations, and words with opposite meaning will keep away from each other in the vector space. To further capture more statistical information, GloVe was introduced [7]. It is a fast and efficient word representation method based on word counts. This model outperformed Word2vec on many tasks in NLP. The latest word embedding model is BERT [8]. BERT generates a prediction for the masked word based on the unmasked words surrounding it. BERT achieved state-of-art performance on a variety of tasks.

However, those methods have a significant disadvantage on sentiment analysis; they ignore information of the sentiment labels. A label in sentiment analysis dataset can be formed by a word that describes the sentiment of sentences or document, e.g., “positive” or “negative”. Label embedding based models, like LEAM [1], can make good use of information in labels, but even LEAM does not perform well on sentiment analysis tasks. Thus, in this paper, we introduce an attention-based method that learns text representations with sentiment lexicons and words simultaneously.

II. RELATED WORKS

A. Word Embeddings

Representing words, sentences and documents is an important task in NLP. Word2vec [4] is an early approach to learn distributed word representations. It learns a vector representation for each word using a neural network language model.

Although typical word embedding methods like Word2vec can capture semantic information from text well, all of them have a fatal flaw when using on text classification tasks; none of them consider using information in the labels.

B. Label Embeddings

Methods of label embeddings are effective in many fields of computer vision. For example, Akata [9] proposed a function that measures the compatibility between an image and a label embedding. By embedding labels, information in the labels can improve the prediction results. In NLP, label embeddings have been proved an effective method on text classification [10]. But there is still a big gap in how to make full use of label embeddings in NLP.

III. LABEL EMBEDDING ATTENTIVE MODEL

Label Embedding Attentive Model(LEAM) proposed by Wang [1] is a text classification model that uses the label embedding framework. Figure 1 shows its framework. Here, \( x \) refers to the input text sequence, \( Y \) are the labels of dataset, \( z \) are the vector representation of \( x \). Furthermore, \( f_o \) is the function to transform words in \( x \) into vectors set \( V \); \( f_1 \) is the function that aggregates word embeddings into text sequence representation \( z \); \( f_2 \) is a classifier that annotate \( z \) with its corresponding label \( y \).

By jointly embedding words and labels in the same latent space, the model can learn text representations using text-label
compatibility. The model outperforms many neural network-based methods at different datasets of text classification, like [11], with much lower model complexity.

To calculate the representation of text, LEAM first embeds words \( x \) and labels \( Y \) into a joint space. This step was done by using a pre-trained word embedding model, like GloVe. Then a word vector matrix \( V \) and a label vector matrix \( C \) are used to measure the compatibility of label-word pairs via cosine similarity.

To further capture the relative spatial information among consecutive words (they called it phrases), they used a masking technique:

\[
\mathbf{u}_l = \text{ReLU}(\mathbf{G}_{p-r:p+r} \mathbf{W}_l + b_l)
\]

Here, \( \mathbf{G}_{p-r:p+r} \) refers to the text phase of length \( 2r+1 \) centered by \( p \), \( \mathbf{W} \) and \( b \) are parameters to be learned. Then the attention score \( \beta \) can be calculated by max-pooling and SoftMax of \( \mathbf{u}_l \) as the following functions:

\[
m_l = \text{max-pooling}(\mathbf{u}_l) \\
\beta = \text{SoftMax}(\mathbf{m})
\]

Here, \( l \) means the \( l \)th element in \( x \). Finally, the text sequence representation can be calculated as:

\[
\mathbf{z} = \sum_l \beta_l \mathbf{v}_l
\]

LEAM learns label-attentive text representations by embedding words and labels into the same space. This technique makes LEAM powerful on text classification tasks. As a result, LEAM has been tested on multiple text classification datasets and reached state-of-art performance on almost every task, except sentiment analysis.

However, LEAM performs not good on sentiment analysis tasks. This may be caused by the lack of label information. For example, suppose learning with the following review: \textit{I was eager to visit this gallery on my most recent trip to Las Vegas}. Because there is no word with strong correspondence with the positive category label \textit{"BEST"}, although this is a positive review, the positive words in review can’t get enough weight for calculation.

**IV. PROPOSED METHOD**

**A. Motivation**

The goal of the proposed method is to improve the performance of LEAM on sentiment analysis by reinforcing connections between words and labels. By sharing the embedding space of words and labels, LEAM can learn a label-attentive text representation. In this representation, applying attention mechanisms, words that have high relevance with labels tend to have high weights in the final result. However, LEAM is less effective in sentiment analysis domains because LEAM only uses the original labels pre-defined by supervised datasets directly. For example, Figure 2 shows the original labels used in Yelp dataset, which has five sentiment categories from BEST to WORST. LEAM only uses them as label information. On the other hand, we are going to take multiple words relevant to the sentiment categories in the proposed method because it is observed that people express the concept of “sentiment” by diverse kinds of words.

Moreover, we found that the original labels are sometimes not suitable for the use of LEAM. For example, the dataset used in LEAM’s experiments, employs the label WORST to represent reviews with 1 star, and BAD to represent reviews with 2 stars. However, SentiWords\(^1\) a well-known repository of sentiment lexicons, states that BAD represents much negative meanings than WORST. Thus, the direct use of the original label may reduce the performance of sentiment analysis. Hence, in order to cast better connections between words and labels, we create a set of categorical lexicons to replace the original labels of dataset.

**B. Categorical Lexicons**

The creation of a set of categorical lexicons aims to better describe specific sentiments instead of the original labels. The set is created based on Sentiwords\(^1\) and English Word Frequency\(^2\). Sentiwords is a sentiment polarity dictionary, words are associated with a sentiment score between -1 and 1.

\(^1\)https://hlt-nlp.fbk.eu/technologies/sentiwords  
\(^2\)https://www.kaggle.com/rtatman/english-word-frequency

---

**Table I**

<table>
<thead>
<tr>
<th>Category</th>
<th>#Lexicons</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORST</td>
<td>96</td>
<td>terrible, awful</td>
</tr>
<tr>
<td>BAD</td>
<td>291</td>
<td>garbage, cheat</td>
</tr>
<tr>
<td>MIDDLE</td>
<td>5,170</td>
<td>same, wide</td>
</tr>
<tr>
<td>GOOD</td>
<td>771</td>
<td>important, dream</td>
</tr>
<tr>
<td>BEST</td>
<td>134</td>
<td>amazing, love</td>
</tr>
<tr>
<td>Total</td>
<td>5,802</td>
<td></td>
</tr>
</tbody>
</table>

---

**Fig. 2. Separation of SentiWords**

---

**Fig. 1. Framework of LEAM**
English Word Frequency is a dataset that contains the counts of commonly-used single words on the English language web. In detail, we firstly apply data reduction on SentiWords according to English Word Frequency because SentiWords includes rarely used words. We select the 10,000 most frequently used words from English Word Frequency dataset. Words that are not in the above selection will be deleted.

Then, because words that have different PoS are assigned with different sentiment score in SentiWords, for those words we only keep 1 sentiment score by averaging their different PoS’s sentiment score. As a result, we reduce the amount of words in SentiWords from 155,000 to 8966.

Then we separate words in SentiWords according to Figure 2. To avoid the word interference between different categories, we use 0.1 as the interval between categories, which means lexicons in different categories have at least 0.1 sentiment score difference. Furthermore, when we use 0.3 sentiment score as the range of word segmentation, we found that the result turned to be too unbalanced. There are very few lexicons in BEST and WORST category, while MIDDLE has many lexicons. So we used an unbalanced separation method to separate words, as Figure 2 shows. This separation allows the number of words in each category to be as even as possible while ensuring that labels of dataset like ‘worst’ and ‘bad’ are in the same category.

After this step, we finally get a set of categorical lexicons with 5802 words. You can see the detail of words in Table 1. We only keep 1 sentiment score by averaging their different sentiment score in SentiWords, for those words are not in the above selection will be deleted.

C. Joint Embeddings

1) Simple averaging: LEAM uses the dataset’s category label itself as label embedding input, which may reduce sentiment analysis performance because of the lack of information in a single label. So we firstly proposed simple averaging method.

The idea of simple averaging is simple: replace the label distribution C in LEAM with E, which is the averaged vector of categorical lexicon set we generated.

For details, for each word \( s \) in categorical lexicons \( S \), we get its word vector \( e_{i,j} \), \((1 \leq i \leq m, 1 \leq j \leq n)\) by a pre-trained word embedding model. Here \((i,j)\) means that \( s \) is the \( i \)th word in \( j \)th category. For example, when experimenting with the Yelp dataset, there are five categories in the dataset, so we take \( n = 5 \), and \( m \) depends on how many lexicons we want to use.

In simple averaging strategy, we get a label vector \( E_j \) by calculating the average vector of \( e_{(i,j)} \) in the \( j \)th category

\[
E_j = \text{Average} \left( \sum_{i=1}^{m} e_{(i,j)} \right). \tag{4}
\]

And we get a new label matrix \( E \) by stacking each \( E_j \). Then we use label matrix \( E \) as same as matrix \( C \) in LEAM. The rest of the procedures remains same as LEAM, except the matrix \( E \) is used instead of \( C \).

2) Attention-based weighted averaging: For better usage of the categorical lexicons, we propose a novel attention-based method( AW ave.) that jointly embeds words and multiple categorical (sentiment) lexicons. Figure 3 shows the framework of the proposed method. Here, as an example, it assumes that we are experimenting on a three-categories-dataset and using three lexicons in categorical lexicon set for each category.

For details, for each sentiment lexicon \( s \) in \( S \), we get its word vector \( e_{i,j}(s) \) as same as simple averaging. Hence we get a set of lexicon matrices \( E \), which has \( n \) matrices, each matrix \( E^j \) has \( m \) vectors.

Then, different from simple averaging method, to calculate the weights for each category separately, we calculate element-wise cosine similarity between \( V \) and \( E^j \) and get a similarity matrix \( B^j \):

\[
B^j = \left( E^j^T V \right) \odot \tilde{B}^j \tag{5}
\]

Here, \( \tilde{B}^j \) refers to the normalization matrix of \( B^j \), and each vector \( b_{(s,j)} \) in \( B^j \) contains \( m \) similarity weights. By averaging those weights, we get averaged matrix set \( G \):

\[
G^j = \text{Average} \left( \sum_{i=1}^{m} b_{(s,j)} \right). \tag{6}
\]

Then, \( \beta^j \) can be calculated as same as equation (1) and (2), by max-pooling and SoftMax of \( u_1 \). And we finally get attention score set \( \beta \) by stacking \( \beta^j \). Here, each \( \beta^j \) represents the weight of the input text sequence in a specific category. By calculating the weighted average, we get the text sequence representation of a specific category \( z_j \):

\[
z_j = \sum \beta^j v_t. \tag{7}
\]

Here, \( z_j \) refers to the sentence representation that is enhanced with the theme of specific category. For example, in a sentiment analysis dataset, \( z_j \) may stand for a positive-sentiment-enhanced sentence representation. Thus, the representation of input text sequence can be calculated by concatenating attention weighted vectors.
\[ z = \text{Concatenate}(z_j). \]  

The core idea of the method is that words in context can have very different compatibility to sentiment lexicons that belong to the same category. In order to make good use of categorical lexicon set, we calculate the attention score between words and every sentiment lexicon. Hence, the compatibility between words and sentiment lexicons can be fully used in the proposed method. Moreover, unlike in LEAM, we calculate \( \beta \) on each category to sharply deliver the text-label compatibility to \( z \).

V. Experiments

A. Setup

a) Dataset: We used the Yelp Review Full dataset generated by Yahoo to test our model. It is obtained from the Yelp Dataset Challenge in 2015\(^3\), the task is sentiment classification of polarity star labels ranging from 1 to 5. Each class of the dataset has one of the five following labels, ‘BEST’, ‘GOOD’, ‘MIDDLE’, ‘BAD’, and ‘WORST’. We split the dataset to 650,000 train reviews and 50,000 test reviews.

b) Parameters: We used the same parameter as the baseline model, LEAM. For details, We used 300 dimensional GloVe pre-trained word embeddings as the input of embedding. The window size \( r \) of the mask was set at 90. The model was optimized by Adam Optimizer, with learning rate 0.001, and a minibatch size of 100.

c) Model set: We used LEAM model as a baseline model and compared its performance with our proposed methods which use 1, 10, 35 and 100 words from categorical lexicons for each category. Thus, we had 8 models to be compared. When using 1, 10, and 35 words, we simply randomly choose word from categorical lexicon set, and we take the average accuracy of five experiments as the final result. We did not apply experiment on AW Ave.(m=1), because AW Ave. needs multiple lexicons. When using 100 words from category set, we choose 100 most frequently used words from BAD, MIDDLE, and GOOD categories, and used all words from WORST and BEST category. For classification function \( f_2 \), we used a SoftMax function, which is the same as the LEAM used. Here, \( y = \text{SoftMax}(z_n') \), with \( z_n' = W_2 z_n + b_2 \), and \( W_2 \) and \( b_2 \) are parameters to be trained.

B. Baselines

To fully evaluate the performance of our model, we compared our model with several sentiment analysis baselines:

a) LEAM: Label Embedding Attentive Model(LEAM) proposed by [1] is a text classification model that jointly embeds words and labels into same space.

b) ULMFiT: Universal Language Model Fine-tuning is a novel model that incorporates position-invariance into RNN. By limiting the distance of information flow in RNN, this model achieved state-of-art performance on several text classification benchmarks.

c) DRNN: Disconnected Recurrent Neural Network [6] is a novel model that incorporates position-invariance into RNN. By limiting the distance of information flow in RNN, this model achieved state-of-art performance on several text classification benchmarks.

d) SVDCNN: Squeezed Very Deep Convolutional Neural Networks [13] is a modified Very Deep Convolutional Neural Networks (VDCNN) model that is much smaller and can fit mobile platforms constraints while keeping performance.

C. Result on Dataset

a) Overall Accuracy: The results are shown in Table 2. Our methods outperformed the baseline method LEAM(both results from [1] and results from of implementation) on Yelp dataset. With the increment of the number of sentiment lexicons, the accuracy also slightly increased. Compared to other state-of-art models like DRNN and ULMFiT, our proposed method have slightly lower accuracy but with fewer parameters and less computation than those methods. Compared with SVDCNN, which is also a small model with few parameters, our model out-performed it with 11.83% better accuracy. Finally, we achieved the best accuracy of 64.99% when using 35 lexicons with AW ave. method. We also applied McNemar’s test between baseline method and other proposed methods. The results of McNemar’s test showed that all the proposed method except Simple Ave.(m=1) had significant difference(\( \alpha < 0.01 \)) between the baseline method.

When comparing two proposed methods, we can see that AW ave. is slightly better than Simple Ave. These results suggest that attention-based weights achieved the appropriate selection of sentiment lexicons for each context.

b) Categorical Accuracy: To further show our proposed method’s increment, we tested categorical accuracy between the baseline method and best-performed proposed methods, as Table III shows.

From Table III we can see that by applying multiple categorical lexicons and attention mechanisms, the accuracy of 3 categories: WORST, BAD, and GOOD, achieved better results than the baseline method. While MIDDLE and BEST’s performance slightly decreased. This result showed that our
### TABLE III
**CATEGORICAL ACCURACY**

<table>
<thead>
<tr>
<th></th>
<th>WORST</th>
<th>BAD</th>
<th>MIDDLE</th>
<th>GOOD</th>
<th>BEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAM</td>
<td>78.92</td>
<td>54.19</td>
<td>54.76</td>
<td>48.25</td>
<td>79.93</td>
</tr>
<tr>
<td>Simple Ave. (m=100)</td>
<td>78.69</td>
<td>58.74</td>
<td>52.68</td>
<td>53.55</td>
<td>78.19</td>
</tr>
<tr>
<td>AW Ave. (m=100)</td>
<td>82.07</td>
<td>59.44</td>
<td>51.11</td>
<td>54.02</td>
<td>78.51</td>
</tr>
</tbody>
</table>

### TABLE IV
**CONFUSION MATRIX OF LEAM**

<table>
<thead>
<tr>
<th></th>
<th>Predicted Classes</th>
<th>WORST</th>
<th>BAD</th>
<th>MIDDLE</th>
<th>GOOD</th>
<th>BEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Classes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WORST</td>
<td>7892</td>
<td>1711</td>
<td>198</td>
<td>76</td>
<td>123</td>
<td></td>
</tr>
<tr>
<td>BAD</td>
<td>2281</td>
<td>5419</td>
<td>1899</td>
<td>264</td>
<td>137</td>
<td></td>
</tr>
<tr>
<td>MIDDLE</td>
<td>362</td>
<td>1807</td>
<td>5476</td>
<td>1894</td>
<td>461</td>
<td></td>
</tr>
<tr>
<td>GOOD</td>
<td>82</td>
<td>178</td>
<td>1579</td>
<td>4825</td>
<td>3336</td>
<td></td>
</tr>
<tr>
<td>BEST</td>
<td>90</td>
<td>57</td>
<td>237</td>
<td>1623</td>
<td>7993</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE V
**CONFUSION MATRIX OF AW Ave.**

<table>
<thead>
<tr>
<th></th>
<th>Predicted Classes</th>
<th>WORST</th>
<th>BAD</th>
<th>MIDDLE</th>
<th>GOOD</th>
<th>BEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Classes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WORST</td>
<td>8207</td>
<td>1578</td>
<td>50</td>
<td>68</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>BAD</td>
<td>2380</td>
<td>5944</td>
<td>1302</td>
<td>273</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>MIDDLE</td>
<td>329</td>
<td>2298</td>
<td>5111</td>
<td>1883</td>
<td>379</td>
<td></td>
</tr>
<tr>
<td>GOOD</td>
<td>88</td>
<td>205</td>
<td>1543</td>
<td>5402</td>
<td>2762</td>
<td></td>
</tr>
<tr>
<td>BEST</td>
<td>113</td>
<td>77</td>
<td>210</td>
<td>1749</td>
<td>7851</td>
<td></td>
</tr>
</tbody>
</table>

The last review are reviews that contain no categorical lexicon. It can be seen that no matter the review contains category lexicon or not, AW Ave. can highlight the expressive words in the review accurately. For example, the word *comfortable* has been highlighted to BEST category.

The first two reviews are reviews that have categorical lexicons inside. For example, *beautifully, perfectly, and awful*. It is clear that our model highlighted words with accurate color.

**D. Representation Ability**

To further demonstrate the ability of AW Ave., we designed a series of experiments to show how AW Ave. Learn a meaningful representation of reviews. Figure 4 shows the result.

In Figure 4, we randomly extracted several correctly classified reviews from the Yelp dataset. We use different colors to highlight words that relevant to different categories. For example, words highlighted with yellow are highly relevant to the MIDDLE category. Moreover, the thicker the color is, the higher weight the word has.

The first two reviews are reviews that have categorical lexicons inside. For example, *beautifully, perfectly, and awful*. It is clear that our model highlighted words with accurate color.

**VI. Conclusions**

In this paper, we proposed an embedding method that jointly learns information of words and sentiment labels simultaneously. The proposed method uses a set of categorical lexicons...
to represent sentiment label information. The attention-based weights served well and finally reached 64.99% accuracy.

For further research, we plan to create a more effective categorical lexicon set. We believe that the decline in GOOD and BEST category accuracy is caused by the categorical lexicons we are using. What is more, we also plan to apply more experiments on datasets in other languages, like Chinese or Japanese, to prove our proposed method’s effectiveness.

REFERENCES


