An Ensemble Model for Sentiment Analysis of Movie Reviews

Shahriar Shayesteh
49495575

Farhad Dashti Ardakani
96223292

Reza Sadoughian
85293141

Abstract

Sentiment analysis is a task of classifying texts based on their meanings. In this work, we are trying to predict the polarity of movie reviews as positive or negative. There are many works on sentiment analysis of movie reviews. This work initially is inspired by a Kaggle competition[1]. In this paper, we introduce an ensemble method that consists of three models. The prediction is based on the weighted average of these three models. The models are chosen based on the three best deep learning architectures on the subject of binary text classification. Experimental results show that our proposed model achieves comparable performance with the state of the art architectures in the task of binary text classification.

1 Introduction

Nowadays, a huge amount of text documents in the form of messages, comments, feedback, and reviews are available on the web. Many forms of information can be extracted from these documents because humans can express their opinion by writing. Therefore, extracting information from texts in order to have a better estimate of people's opinions about a specific subject is a very common task. In this paper, we focus on the task of sentiment analysis of movie reviews to classify people's opinions as positive or negative reviews. In sentiment analysis task, to capture the sense of a sentence, we need to consider the dependencies between the words as well as meaning of each single word. Thus, after generating vector representations of the words, we use a proper binary classifier model to be able to learn the semantics of reviews efficiently and to predict the sense of new movie reviews correctly.

In this paper, we propose an ensemble averaging method consisting of three models. Each model tries to address this task from a different point of view. Models considered in this architecture are: convolutional neural network (CNN) architecture proposed by Kim [2014], CNN followed by a recurrent neural network (RNN) which is called CNN-RNN [Hassan and Mahmood, 2018] and bidirectional long short term memory (Bi-LSTM) architecture introduced by Minaee et al. [2019]. CNN architecture using a max-pooling layer extracts high-level features in the text that are invariant to local translation [Minaee et al., 2019]. CNN-RNN architecture is able to capture the long dependencies in the data [Hassan and Mahmood, 2018]. Bi-LSTM architecture can capture temporal correlation and dependencies in the text from the future and the past [Minaee et al., 2019]. At the prediction stage, we use the weighted average of the predictions made by these models to obtain the final prediction for our proposed ensemble model. To do weighted averaging, there is a heuristic method to obtain the weights for each model. Each weight is determined based on the ratio of the training accuracy of each model on the total accuracy of the models.

[1] Bag of Words Meets Bags of Popcorn

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Although employing deep learning structures are effective in capturing semantics of a text, there are some cases that semantics are not directly expressed in the text, e.g., expressions, collocations or sarcastic sentences. In these cases, our model along with other proposed deep learning models cannot achieve very satisfactory results since they are not able to fully extract deeper layers of the meanings. However, averaging three different novel models can leverage the power of each novel model in order to have a better performance not only in general cases but also on these special cases.

In this paper, we want to improve the performance of existing models on the task of sentiment analysis by averaging three models proposed by Minaee et al. [2019], Kim [2014], Hassan and Mahmood [2018]. We use two popular datasets in the task of sentiment analysis which are IMDB movie reviews[2] and SST datasets[3].

2 Related Work

Before the advent of neural network, words were treated as discrete and atomic symbols. In this manner, the models were able to capture just the relationship between the individual symbols. Therefore, the amount of information could be extracted from sentences by these methods was not too much [Mikolov et al., 2013]. One of the most common classical models for doing the task of sentiment analysis was using bag-of-words to represent each sentence. A classic classifier such as naive Bayes or logistic regression was employed to train the bag-of-words [McCallum et al., 1998, Socher et al., 2011]. The main problem with the bag-of-words approach is ignoring the information about semantics and word ordering in a sentence [Socher et al., 2011, Collobert and Weston, 2008]. Furthermore, using linear classifiers that are not able to capture dependent structures between features and classes is another drawback of the classical models. In other words, they are not able to generalize the problem properly.

In recent years, after the significant performance of deep neural networks (DNNs) in the field of computer vision [Krizhevsky et al., 2012, Collobert et al., 2011], they have been used in the field of natural language processing (NLP). One of the greatest achievements of neural networks in sentiment analysis tasks is to model the words in the text as vectors. The vector representation of words using neural networks can capture semantic and syntactic information of the words. Words with similar meanings have also similar word vectors in the vector space. As a result, continuous bag of words (CBOW) and continuous skip-gram as log-linear models are introduced by Mikolov et al. [2013]. Both models are suffering from slowness in the training time. Consequently, negative sampling is introduced by Pennington et al. [2014] as an alternative model to these models. Negative sampling improves not only the training time but also the efficiency of the word vector representations. Vector representation models by Mikolov et al. [2013] and Pennington et al. [2014] which are commonly called word2vec, revolutionized the field of NLP. However, they are not able to come up with a more generalized vector representation for the words due to considering a small window around each word. In hindsight, global vectors (GloVe) for word representation are proposed by Pennington et al. [2014] which reuses co-occurrence matrices within a fixed-size window.

In the terms of classifying, deep learning models also perform better than classical classifiers. CNN architecture which was introduced by Kim [2014] improved the accuracy in sentiment analysis tasks surprisingly. This architecture is also used for aspect-based sentiment analysis to classify twitter comments to three classes of positive, negative and neutral [Severyn and Moschitti, 2015]. However, this architecture suffers from loss of information due to using pooling layers. Adding LSTM to CNN by Hassan and Mahmood [2018] instead of using a max-pooling layer was an effective way to remedy some of the problems with CNN architecture with max pooling. CNN-RNN model is also able to capture some long-term dependencies in the data. Furthermore, recursive neural conditional random fields have been used for aspect-based sentiment analysis [Wang et al., 2016]. This architecture suffers from inaccuracy and slowness on noisy data.

In this work, we are inspired by Minaee et al. [2019] employing an ensemble method which helps us to boost the performance of the model on the task of sentiment analysis. In this paper, we employ a weighted ensemble method that takes advantage of models proposed by Minaee et al. [2019], Kim [2014], Hassan and Mahmood [2018] on the task of sentiment analysis. In the next section, we introduce our proposed model.

[2] Stanford Large Movie Review Dataset

[3] Stanford Sentiment Treebank Dataset

3 Model

In this section, our proposed model is presented. Our architecture is a weighted average ensemble of three models proposed by Minaee et al. [2019], Kim [2014], Hassan and Mahmood [2018]. In this work, we try to leverage the power of each model in classifying the movie reviews to have a better classifier overall. In this section, first we describe the existing techniques used for sentiment analysis of movie reviews. We explain word embedding technique that we are using in this paper. We also describe existing models for classification such as CNN architecture, LSTM and Bi-LSTM networks. At the end, we describe our proposed model which is an ensemble of three existing models.

3.1 Existing Techniques and Models

3.1.1 Embedded Layer

To transform input words into feature vectors, we use pre-trained GloVe word embeddings to capture semantic and syntactic information in the words. The pre-trained GloVe word embeddings were trained on a dataset consisting of one billion tokens (words) with a vocabulary of 400 thousand words. Our input data is a sequence of words of length L. By feeding the sequence of input words into the embedding layer, the output is a matrix of L by d where d is the embedding dimension which is set by us.

3.1.2 The Convolutional Layer

In this paper, we use CNN structure proposed by Kim [2014]. Let the i^{th} word be $\mathbf{x}_i \in \mathbb{R}^d$ where d is the dimension of the word vector for each word. Since reviews are the input of the network, words in each review are concatenated by row to form an input vector. Here we consider a fixed-length review of L. Therefore, our input is a vector of length Ld [Kim, 2014]:

$$\mathbf{x}_{1:L} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \oplus \mathbf{x}_L, \tag{1}$$

where $\oplus$ is a concatenation operator and $\mathbf{x}_{i:i+j}$ means vector $\mathbf{x}_i$ to vector $\mathbf{x}_{i+j}$ are concatenated together .

To perform text classification with CNN, after embedding different words of a review which are stacked together to form a vector, convolution filters of different lengths are applied to input data to produce a new feature representation. We represent a filter as $\mathbf{w} \in \mathbb{R}^{ld}$ where $l \in \{1, \dots, L\}$. For example, a feature map c_i is generated by convolution of a filter by a window of words with length l plus bias term $b \in \mathbb{R}$ [Kim, 2014]

$$c_i = f(\mathbf{w} \circledast \mathbf{x}_{i:i+l-1} + b), \tag{2}$$

where $\circledast$ is convolution operator and f is a nonlinear function such as the Relu function. After applying this filter to each possible window of words in the sentence $\{\mathbf{x}_{1:l}, \mathbf{x}_{2:l+1}, \dots, \mathbf{x}_{L-l+1:L}\}$, a feature map corresponding to a filter is produced as follows [Kim, 2014]:

$$\mathbf{c} = [c_1, c_2, \dots, c_{L-l+1}], \tag{3}$$

where $\mathbf{c} \in \mathbb{R}^{L-l+1}$. The next step is to feed these feature maps to either a max-pooling layer or LSTM network.

3.1.3 Max Pooling Layer

Max pooling layer is introduced by Collobert et al. [2011]. We apply it over the feature maps to take a maximum value $\tilde{c} = \max \mathbf{c}$ to represent the most important feature corresponding to each feature map. CNN with max pooling layer is shown in Figure 1.

3.1.4 The Recurrent Layer

One type of neural networks used in sequence modeling are RNNs. At each time step t, the input vector $\mathbf{x}_t$ with legth n and hidden state from the previous time step $\mathbf{h}_{t-1}$ are fed into the recurrent layer, and then the recursive operation is applied on them to form the current time step hidden layer as follows:

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}), \tag{4}$$

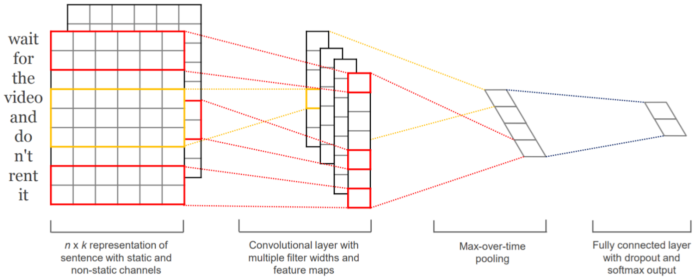
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1: CNN with max pooling layer [Kim, 2014]

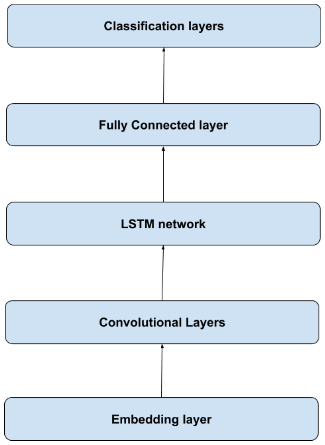

Figure 2: CNN+LSTM (CNN-RNN) model

where $\mathbf{W} \in \mathbb{R}^{h \times n}$, $\mathbf{U} \in \mathbb{R}^{h \times h}$, $\mathbf{b} \in \mathbb{R}^{h}$ are parameters and f is an element-wise non linearity. Vanishing gradient and exploding gradient are the reasons to not to use vanilla RNN anymore [Maas et al., 2011]. One of the models that can address aforementioned drawbacks associated with vanilla RNN is the long short-term memory (LSTM) [Boden, 2002]. The LSTM architecture contains a memory cell which takes input data $\mathbf{x}_t$, hidden layer and memory cell of previous time step $\mathbf{h}_{t-1}$, and $\mathbf{c}_{t-1}$ and produces $\mathbf{h}_t$ and $\mathbf{c}_t$ as follow [Minaee et al., 2019]:

$$\begin{aligned}
i_t &= \sigma(\mathbf{W}^i \mathbf{x}_t + \mathbf{U}^i \mathbf{h}_{t-1} + \mathbf{b}^i), & (5)\\
f_t &= \sigma(\mathbf{W}^f \mathbf{x}_t + \mathbf{U}^f \mathbf{h}_{t-1} + \mathbf{b}^f), & (6)\\
o_t &= \sigma(\mathbf{W}^o \mathbf{x}_t + \mathbf{U}^o \mathbf{h}_{t-1} + \mathbf{b}^o), & (7)\\
g_t &= \sigma(\mathbf{W}^g \mathbf{x}_t + \mathbf{U}^g \mathbf{h}_{t-1} + \mathbf{b}^g), & (8)\\
\mathbf{c}_t &= (f_t \odot \mathbf{c}_{t-1} + i_t \odot g_t), & (9)\\
\mathbf{h}_t &= o_t \odot \tanh(\mathbf{c}_t), & (10)
\end{aligned}$$

where $\tanh$ and σ are hyperbolic tangent and the element-wise sigmoid functions, i_t is input gate, f_t and o_t are referred to as forget and output gates. At the first time step, h_0 and c_0 are initialized to zero vectors. The element-wise multiplication operator is illustrated with $\odot$.

LSTM outperforms vanilla RNNs on many tasks [Sundermeyer et al., 2015]. LSTM network on top of the CNN architecture as an alternative to the max-pooling has been proposed by Hassan and Mahmood [2018]. We use this model as a part of our ensemble model. Figure 2 shows the structure of CNN+LSTM model.

3.1.5 Bi-LSTM

In many applications, we need temporal information flow in both directions. To address this issue, the Bi-LSTM network is introduced in Minaee et al. [2019], which consists of two hidden layers on the input sequence. The first hidden layer works the same as the LSTM network while the second hidden layer is on the reversed copy of the input sequence. In this manner, by considering both past and future information, the process of learning is faster and more efficient.

In this paper, we use a one-layer bi-LSTM network that directly gets the output of the embedding layer and predicts the sentiment for that. Figure 3 demonstrates the model that we use in this paper.

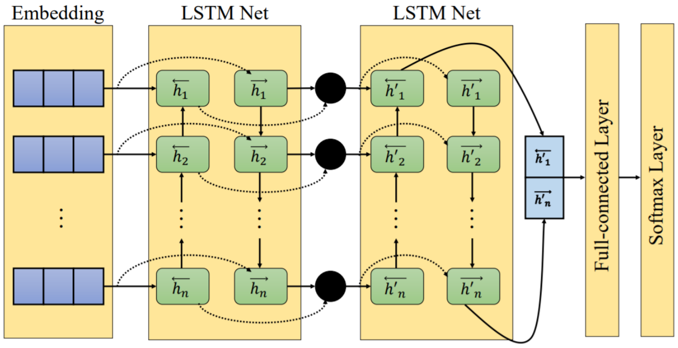

Figure 3: Bi-LSTM network [Minaee et al., 2019]

3.1.6 The Classification layer

We need to predict the sentiment in the input data at the end. As a result, the classification layer is a logistic regression that predicts the polarity of the input. The input data to the classification layer is the output of the max pooling or LSTM network after feeding into a fully connected layer. The logistic regression classifier is a softmax classifier while the number of the classes is two. The following formula shows a softmax classifier which computes the predictive probabilities for all k classes [Socher et al., 2011]:

$$p(y = k \mid X) = \frac{\exp(w_k^T x + b_k)}{\sum_{c=1}^{k} \exp(w_c^T x + b_c)}, \tag{11}$$

where w_k and b_k are weight vectors and bias vectors, respectively.

3.2 Our Model

As it was mentioned before, our proposed model is a weighted average ensemble model consisting of three novel architectures used in the task of sentiment analysis [Minaee et al., 2019, Kim, 2014, Hassan and Mahmood, 2018]. Figure 4 shows the architecture of our proposed model. In summary, CNN architecture followed by a max-pooling layer in Minaee et al. [2019] is able to leverage relative information shared between words in a sentence. CNN architecture followed by an LSTM network (CNN-RNN) is able to capture long-term dependencies in the data. The bi-LSTM network can address other aspects of this problem which was not answered before by considering past and future information in the input sequence. We believe that we can boost the performance of sentiment analysis on the movie reviews by putting these three models together.

In our work, we perform a weighted average ensemble on three models to predict the sentiment of each review. The weight of each model is set based on their performance in the training phase. Each weight is equal to the accuracy of each model on the validation set over the total accuracy of the models. Consider CNN + max pooling as model 1, Bi-LSTM network as model 2 and CNN + LSTM (CNN-RNN) as model 3. The weights corresponding to each model is as follow:

$$\alpha_i = \frac{\text{acc}_i}{\sum_{j=1}^{3} \text{acc}_j}, \; i = 1, 2, 3, \tag{12}$$

where acc_i is the accuracy of model i on the validation set. The prediction of our model is obtained by

$$\text{pred} = \frac{\alpha_1 \text{pred}_1 + \alpha_2 \text{pred}_2 + \alpha_3 \text{pred}_3}{\alpha_1 + \alpha_2 + \alpha_3}, \tag{13}$$

where $\text{pred}_i, i = 1, 2, 3$ is the prediction of the sentiment of the movie review made by model i.

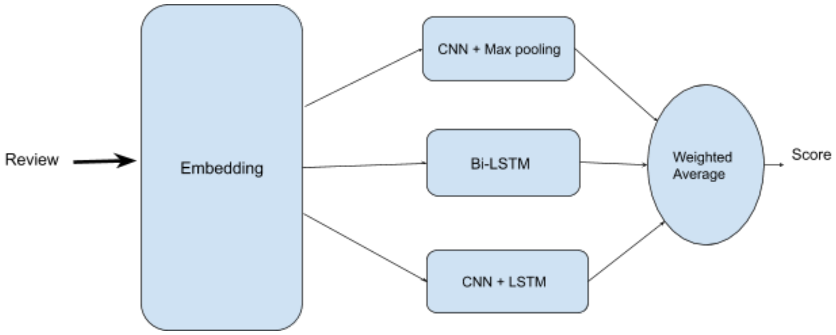

Figure 4: Our proposed model architecture.

4 Datasets and Experimental Setups

In this section, we describe datasets and settings that we are using in our simulations. We evaluate the performance of our proposed model on two popular datasets in the task of sentiment analysis: the Stanford Large Movie Review dataset (IMDB) and the Stanford Sentiment Treebank dataset (SSTb) [Socher et al., 2013]. These datasets are derived from Rotten Tomatoes movie reviews [Pang and Lee, 2005].

4.1 Stanford Large Movie Review Dataset (IMDB)

The IMDB dataset is the most common benchmark for sentiment analysis. This dataset is introduced in Maas et al. [2011]. IMDB dataset contains 5000 labeled binary reviews and it has a balanced distribution for each label. It is important to note that each review in this dataset has several sentences. In this paper, we use half of the data as training data and the other half as testing data. We use 20% of the training data as the validation set.

4.2 Stanford Sentiment Treebank Dataset (SSTb)

SSTb is another significant dataset in sentiment analysis tasks. This dataset is proposed in Socher et al. [2013], Pang and Lee [2005]. SSTb dataset contains 11,855 reviews. These reviews are taken from the Rotten Tomatoes website and unlike the IMDB, each review is one sentence. SSTb dataset has three subsets: 8544 sentences for training, 1101 sentences for validation, and 2201 sentences for testing.

4.3 Hyperparameters and Training

In this work, we use Python aspr ogramming language. Pytorch, Torchtext and spacy are the main frameworks that we are using. We use torchtext and spacy to preprocess the data and Pytorch to build our model.

Pre-trained GloVe word embeddings with 100 dimensions are employed as the word embedding technique in this paper. Moreover, we use padding to have a fixed length input data. Our input length is set to 1300.

CNN architecture utilizes rectified linear units as nonlinear function. CNN has one hidden layer, filter windows (l) of 1, 2, 3, 4, 5 with 100 feature maps each and dropout rate (p) of 0.5. In CNN plus max-pooling layer, one layer of max-pooling is applied on the feature maps of the CNN. In CNN-RNN architecture, one layer LSTM network is employed. Hidden and cell units have also 32

Table 1: Classification Accuracy of Models on IMDB Dataset

Model	Classification Test Accuracy
CNN	82.30%
CNN-RNN (CNN + LSTM)	86.41%
Ensemble of CNN and Bi-LSTM	87.10%
Ensemble of CNN, Bi-LSTM and CNN-RNN	**87.84%**

Table 2: Classification Accuracy of Models on SST Dataset

Model	Classification Test Accuracy
CNN	79.23%
CNN-RNN (CNN + LSTM)	74.30%
Ensemble of CNN and Bi-LSTM	79.28%
Ensemble of CNN, Bi-LSTM and CNN-RNN	78.84%

nodes. For Bi-LSTM architecture, we utilize a one layer Bi-LSTM network. The size of the hidden layer is also 32.

In this paper, we use Adam optimizer with a learning rate of 0.001 for each model. Moreover, the Batch size is 32 for all training and testing data. Finally, we validate our model over 50 epochs. A value for the best number of epochs is chosen based on the loss function, denoted as N_b. At the end we train and test our model over N_b epochs.

5 Results and Discussion

In this section, we report the classification accuracy of models discussed beforehand and our proposed model. We train the models on both IMDB and SST dataset. Table 1 contains the classification accuracies on IMDB dataset. Our model which is an ensemble of three existing models, has a performance gain compared to each individual model. This is because our model considers three different models. Then it predicts the sentiment of each review based on the weighted average of predictions made by these three models. Actually, our model takes advantage of each of three models.

Table 2 contains the accuracies of different models on SST dataset. For STT dataset, the performance of our model is a little bit lower than the model which is ensemble of CNN and Bi-LSTM model. Since CNN-RNN model does not have high accuracy on sentiment analysis on SST dataset, it degrades the performance of our model in making predictions.

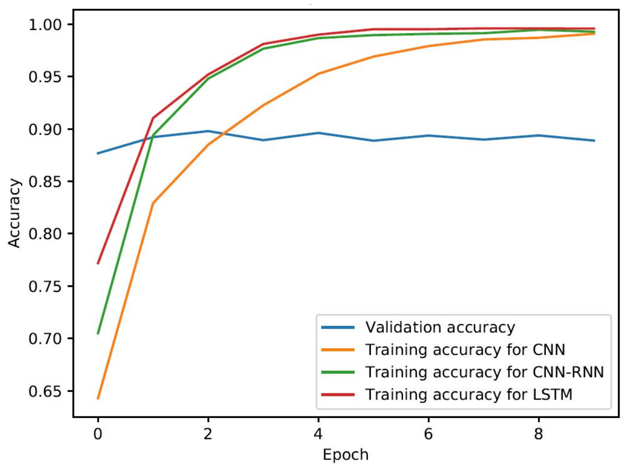

Figure 5: Training accuracy versus epoch for different models

Figure 5 shows the training accuracy of three models (ensembled by our proposed model) versus the number of epochs on the dataset.

6 Conclusion

In this work, we introduced our ensemble model for sentiment analysis by weighted averaging on three novel models as mentioned before. Combining different models with different abilities to capture the sentiment of the reviews gives us this power to extract the sentiment of each review more accurately. Although our model has a performance gain compared to other models, the training time for our ensemble model is almost 13 times higher than CNNs and approximately 4 times higher than CNN-RNN for each epoch. In future work, we can utilize a new method of pre-training language representations developed by Google called bidirectional encoder representations for transformers (BERT) [Devlin et al., 2018]. BERT can replace the embedding layer. If we can tune the BERT very well, then it can lead to faster training and better performance.

References

M. Boden. A guide to recurrent neural networks and backpropagation. *the Dallas project*, 2002.

R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

A. Hassan and A. Mahmood. Convolutional recurrent deep learning model for sentence classification. *IEEE Access*, 6:13949–13957, 2018.

Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

A. McCallum, K. Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

S. Minaee, E. Azimi, and A. Abdolrashidi. Deep-sentiment: Sentiment analysis using ensemble of CNN and Bi-LSTM models. *arXiv preprint arXiv:1904.04206*, 2019.

B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

J. Pennington, R. Socher, and C. D. 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.

A. Severyn and A. Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962, 2015.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics, 2011.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

M. Sundermeyer, H. Ney, and R. Schlüter. From feedforward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3): 517–529, 2015.

W. Wang, S. J. Pan, D. Dahlmeier, and X. Xiao. Recursive neural conditional random fields for aspect-based sentiment analysis. *arXiv preprint arXiv:1603.06679*, 2016.