Sentiment Analysis using Deep Learning on Persian Texts

Behnam Roshanfekr
Master of Science Student
Computer Engineering and IT
Amirkabir University of Technology
Tehran, Iran
b.roshanfekr@aut.ac.ir

Shahram Khadivi*
Adjunct Professor
Computer Engineering and IT
Amirkabir University of Technology
Tehran, Iran
khadivi@aut.ac.ir

Mohammad Rahmati
Associate Professor
Computer Engineering and IT
Amirkabir University of Technology
Tehran, Iran
rahmati@aut.ac.ir

Abstract— **Given the growth rate in the volume of text data and information, text classification has become more practical and handy. Sentiment analysis is one of the text classification applications which can be used in some cases to evaluate products, make market decisions or measure consumer confidence. Most of the methods proposed for this task have concentrated on the English language whereas there have been a few attempts for other languages such as Persian. There are some challenges in Persian Language. For example, it has a wide variety of suffixes. Recently, deep learning approaches have been successfully applied in a variety of NLP applications. Our goal is to evaluate deep learning methods in the Persian language. It can be shown that some of the challenging issues will be addressed when using deep learning methods. We also introduce a dataset of reviews about electronic products in Persian language and evaluate the models on it.**

Keywords-Sentiment analysis; document classification; deep learning; skip-gram model; convolutional neural network; bidirectional LSTM

I. INTRODUCTION

Sentiment analysis deals with constructing instructions to approximate concept which concerns with text classification applications which is a fundamental task in Natural Language Processing. The goal is to classify sentences or documents according to sentiments, attitudes, emotions, opinions and evaluations expressed in them [1]. Due to the growth of information and reviews in the form of comments on the internet, it is necessary to classify them automatically, which could be used in some cases such as product evaluations, marketing decision making or consumer confidence measurement [1][2].

There are three main levels to investigate sentiment analysis. In *document-level*, the task is to determine that document has a positive or negative sentiment [2]. In *sentence-level*, the task is similar to document-level, except that instead of documents, for each sentence determines whether it expresses a positive, negative or neutral opinion. Neutral means sentence expresses no opinion [2]. In *entity-level*, it assumes that for each opinion there is a sentiment and a target. For example, in the sentence "*although the actors of the movie are first grade, I didn't like it*" there is a positive sentiment about actors and a negative sentiment about movie [1][2].

Most approaches, proposed for sentiment analysis, use traditional machine learning techniques which rely on feature engineering. Therefore features extracted from documents can play an important role in classification performance [3]. For this reason, most methods focus on extracting proper features to obtain accurate classification performance [3]. Some of the usual machine learning algorithms used for this goal, are logistic regression, Naïve Bayes, and SVM. Since these algorithms use fixed-length feature vectors, documents should be represented as fixed-length feature vectors [4]. One way is to represent each document as a bag-of-words which is simple and efficient but ignores word orders in documents [4]. This issue can cause some errors in sentiment classification since it's possible for two sentences made of the same set of words, but express different sentiments [9]. Bag-of-words representation also doesn't consider the semantic similarity between words [4].

Most of these methods rely on English language whereas there are a few methods working on other languages such as Persian. Furthermore, Persian is considered a demanding language as a result of its attributes. For example, it uses a wide variety of suffixes, as in sentences "من می‌روم." and "تو می‌روی." two different suffixes are used. It also has different kinds of writing styles such as different writing forms for a specific word [5]. Another problem in sentiment analysis on Persian language is that the available datasets are small, which contains about 1500 labeled documents.

In this paper, to deal with some of the problems proposed for Persian language, we tried to employ two types of deep neural network models to classify text documents according to their sentiment polarity. Recently these deep learning models have achieved noticeable results in different applications such as computer vision, speech recognition, and natural language processing [6][7]. Instead of representing each word using its index in the vocabulary, we used a vector space model (VSM) which was trained on a corpus of reviews in an unsupervised way. It learns a representation for each word in a way that similar words have similar representations. For training the models we also introduced a dataset of about 200761 reviews which

*This work has been done when Shahram Khadivi was with Amirkabir University of Technology.

978-1-5090-5963-8/17/$31.00 ©2017 IEEE

contains about 50000 labeled data. We compare our results with the NBSVM-bi model proposed in [10].

In the reminder of this paper, section 2 describes some of the previous works on sentiment analysis. Section 3 describes the models we used in the process of sentiment classification of Persian reviews. In section 4 we described our dataset, training phases, and our experimental results in detail. Section 5 also includes a conclusion.

II. Related Works

There are a variety of methods, proposed for sentiment analysis. Some of them are based on computational linguistic but most of them are based on machine learning and follow Pang et al. [8], which considered sentiment classification as a kind of text classification and used three supervised machine learning methods including Naïve Bayes, Maximum Entropy and SVM [9]. It's common to use bags-of-words representation for text documents in machine learning based methods. Since this kind of representation doesn't consider the order of the words, it cannot capture complex meanings from documents.

Wang and Manning [10] employed word bigram features in sentiment analysis task. They also proposed a variant of SVM which used log count ratios as feature values called NBSVM [10].

The deep learning approach, as a new field in machine learning, has attracted many researchers to employ it in different applications. Nowadays it appears in a lot of Natural language processing tasks such as sentiment analysis, machine translation and etc [11]. In these methods, it is common to represent words with one-hot vectors which fail to capture the relational structure of lexicon [12]. An advanced representation, encodes word similarities as a kind of distance, in a continuous high-dimensional space. For example Mass et al. [12] introduced a model which captures semantic and sentiment similarities among words. The semantic part of the model learns word vectors using a probabilistic model of documents in an unsupervised way. To capture sentiment similarities it uses a supervised learning. Actually, in this part, the model predicts the sentiment expressed in the context in which the words appear. This causes similar sentiment words to have similar vector representations [12].

Socher et al. [9] introduced a model based on semi-supervised, recursive autoencoders called Recursive Neural network (RNN). He used continuous word vectors as input and Instead of using bag-of-words representation, he employed a hierarchical structure in which can find vector representations for variable-sized phrases. For this goal, the model uses a binary tree in which the leaf nodes correspond to a word vector. Then it computes parent vectors in a bottom-up fashion using a linear compositional function. At the end, it applies a softmax function on the vector representation of phrases to compute a probability distribution over labels. Another model which is called Matrix-Vector RNN (MV-RNN) is introduced in [13]. This model is similar to RNN except that, to represent words and phrases in the binary parse tree, it uses both a vector and a matrix. But one main problem with this model is that it has a large number of parameters which depend on the size of the vocabulary [14]. To solve this problem Socher et al. [14] proposed a model called Recursive Neural Tensor Network (RNTN). This model is also similar to VM-RNN model except that instead of a matrix representation for each word and phrase, it uses a same, tensor-based composition function for all nodes in the binary parse tree.

Le and Mikolov [4] proposed a model called Paragraph-Vector which learns continuous distributed vector representations for variable-length documents in an unsupervised way. These vectors can employ in different applications of text classification.

In this paper, we employed two deep neural network architectures to classify Persian reviews which are about electronic products, depend on the sentiment they express. We collect our data using a crawler from the website www.digikala.com which containing a total of 200761 customer reviews. About 50000 of them have positive or negative labels. We have two learning phase in our models. The first phase is learning vector representations of words using a skip-gram model, proposed in [15], in an unsupervised way. The second phase is learning document sentiments using deep neural networks in a supervised way.

III. Model Description

In this section, we describe the structure of models that we used for sentiment classification.

A. Skip-gram model

The architecture of a Skip-gram model is depicted in Fig. 1. This model is used for learning vector representations of words from a large amounts of text data, in an unsupervised way [15].

It tries to learn a representation vector for a word by predicting its surrounding words in training documents. More formally if we assume that there is a sequence of training words $w_1, w_2, \dots, w_T$, the Skip-gram model tries to maximize the average log probability in (1) [15].

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0}\log p(w_{t+j}|w_t), \quad (1)$$

where c is the context window size around the center word in which, the model tries to predict surrounding words. In a basic formulation, the conditional probability proposed in (1) is computed using the softmax function as showed in (2) [15].

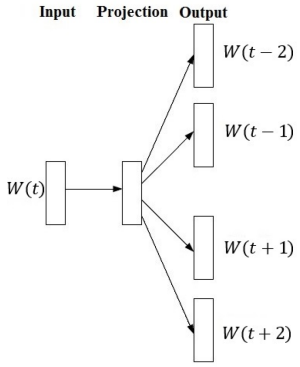

Figure 1. The architecture of the Skip-gram model [15].

$$p(w_O|w_I) = \frac{\exp(v'^T_{wo} v_{wI})}{\sum_{w=1}^{W} \exp(v'^T_{w} v_{wI})}, \quad (2)$$

where W is the vocabulary size. v_w and v'_w are also two kind of vector representations for word w. v_w is the input representation and v'_w is the output representation [15]. Maximizing the conditional probability in (2), means minimizing the cosine distance between the two word vectors. As a result after training, similar words will have similar representation vectors. Another interesting thing that will happened is that, the learned word vectors encode some linguistic patterns which can be shown in the form of linear transformations [15]. In Fig. 2 we visualized some of the learned word vectors. We did this by projecting word vectors down to 2 dimensions using PCA (Principal Component Analysis) dimensionality reduction technique. By inspecting these visualizations, it is somehow apparent that the vectors capture some useful semantic information about words and their relationship to each other. As depicted in Fig. 2, similar words such as "سونی", "اپل" and "هوا وی" (all of them are cell phone brands) have similar vectors. There are also other relationships between word vectors. As an example, Mikolov et al. in [15] explained a relationship between countries and their capitals. This explains why these vectors should be useful as representations of words for NLP tasks such as sentiment analysis.

B. *Bidirectional Long Short Term Memory (LSTM)*

The structure of the LSTM networks is similar to Recurrent Neural Networks [7], except that instead of each self-connected hidden unit there is a kind of memory unit. Despite that the Recurrent Neural Networks are designed to deal with variable lengths sequences, there are some limitations. Actually, the influence of an input sequence would decay or blow up exponentially while circulating around the hidden states. This causes Recurrent Neural Networks not to learn long-term dependencies which is known as vanishing or exploding gradient problem [16].

The LSTM network solves this problem using memory units. Equations (3) to (8) show the process of computing the output of a LSTM hidden unit h_t given input x_t [17].

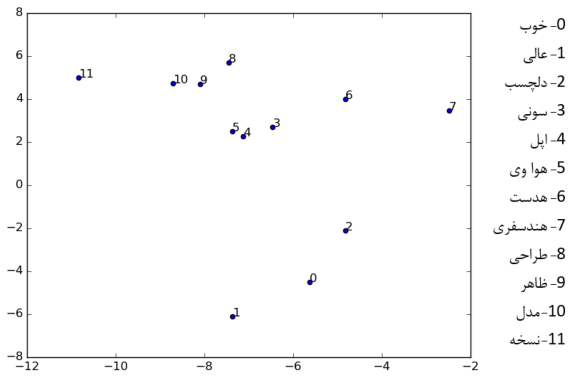

Figure 2. Representation of some learned vectors of Persian words in 2D space after applying PCA

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (3)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (4)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (5)$$

$$g_t = tanh(W_g x_t + U_g h_{t-1} + b_g) \quad (6)$$

$$C_t = i_t * g_t + f_t * C_{t-1} \quad (7)$$

$$h_t = o_t * \tanh(C_t) \quad (8)$$

Where i, o and f are respectively input, output and forget gates. x_t is the memory unit input at time t. g is an intermediate hidden state, C represents the hidden unit internal memory. σ is the logistic sigmoid function. W, U, b are also weight matrices and bias vectors. Notice that in Recurrent Neural Networks the hidden unit computed as $h_t = \tanh(Wx_t + Uh_{t-1})$. Therefore comparing LSTM to Recurrent Neural Networks, in LSTM the hidden unit is computed in a complicated way. Actually this mechanism allows LSTM to store information over long periods of time, which makes it possible to deal with long-term dependencies [17] [16].

The LSTM network explained above only considers the past sequence in calculating the output of a hidden unit. In our classification task where the whole sentence is given, it will be helpful to exploit future sequences. For this reason, we used a bidirectional structure of LSTM. As illustrated in Fig. 3 it involves of two separate hidden units, one computes the output for each sequence in a forward manner ($\vec{h}$) and one computes the output for each sequence in a backward manner ($\overleftarrow{h}$). The corresponding output sequences are concatenated to each other and a mean pooling applies to get the document representation from output sequences. The document representation v, is a high level representation which involves a summary information of document and it can be used as features for classifying the documents according to their sentiment [16][17].

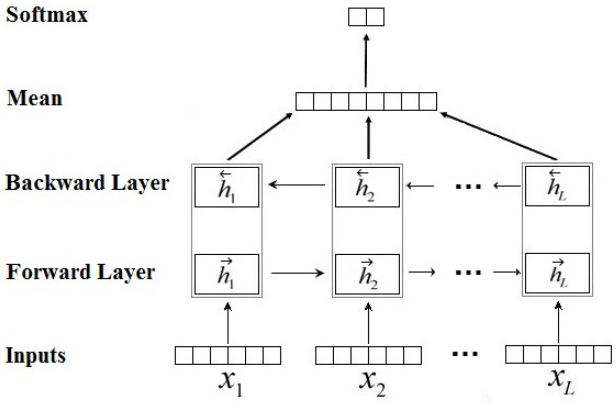

Figure 3. Bidirectional LSTM Network for document classification

After computing the document representation vector v, a softmax layer as in (9) can be used to get a probability distribution over the labels.

$$p = softmax(W_c v + b_c) \quad (9)$$

One solution to deal with overfitting is to apply a 'dropout' to the model. This is another mean of regularization which prevents co-adaptation of hidden units [6].

The dropout operator randomly sets hidden unit values to zero with probability of p. In this way there is a kind of force for intermediate computations to be robust. For LSTMs, [17] suggests to apply the dropout operator only to the non-recurrent connections. In our experiments, we apply the dropout operator to the outputs of each LSTM unit.

C. Convolutional Neural Network (CNN)

The CNN architecture which is used to classify documents upon their sentiment is depicted in Fig. 4. In this setting input documents are represented as a matrix, each row of it corresponds to one token. Typically a token can be considered as a word. Therefore in each row there is a vector, representing a token. Considering a document, has maximum N tokens (We use padding strategy for documents shorter than N tokens.) and each token represents with a d dimension vector, the document will be represented with a matrix $A \in \mathbb{R}^{N \times d}$. In this way each document can be treated as an image. Therefore a convolution operation can be performed on a window of h tokens via linear filters. Since each row of the document matrix represent a token, it makes sense to use filters in which their widths are equal to the dimensionality of the word vectors [6][18].

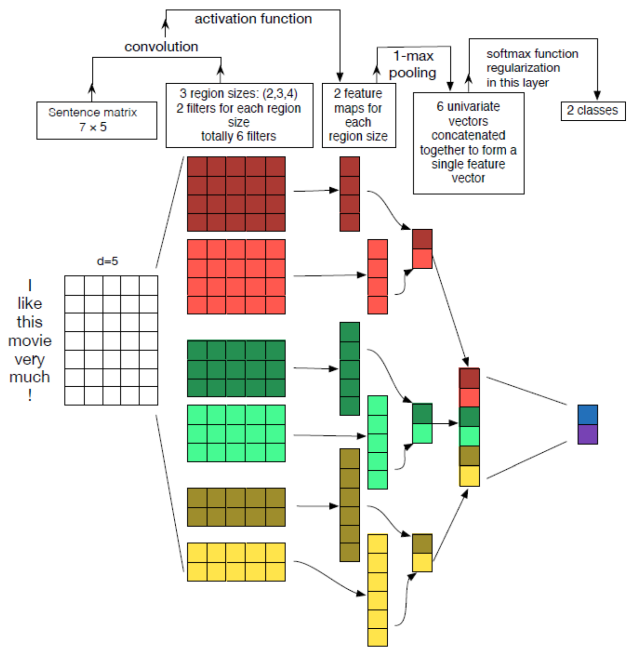

Figure 4. CNN architecture for document classification [18]

Therefore, to perform a convolution operation on h words a filter $w \in \mathbb{R}^{h \times d}$ should be involved. Let $A[i:j]$ be a window of tokens from token i to token j of the document matrix A [6][18]. A feature c_i is calculated based on a token window of size h by

$$c_i = f(w \cdot A[i:i+h-1] + b), \quad (10)$$

where $i = 1 \dots N - h + 1$, $w \in \mathbb{R}^{h \times d}$ is the filter weight matrix, $b \in \mathbb{R}$ is the bias term, f is an activation function and $\cdot$ is the dot product [6][18]. After applying filter w to each possible word window and putting calculated features together in a vector, there would be a feature map such as

$$c = [c_1, c_2, \dots, c_{N-h+1}], \quad (11)$$

where $c \in \mathbb{R}^{N-h+1}$. The size of the feature map depends on the size of the document and the size of the filter, which has been applied to it. Therefore a pooling function is needed to get a fix-length vector. This architecture uses a max pool function which extracts the maximum value from each feature map as the feature correspond to the particular filter that has been applied. One may apply different numbers of filters in different sizes. So in this way for each filter, one feature value will be computed. This solves the problem of variable length documents, since for each document, there are a fix number of feature values equal to the number of filters applied in convolution step. Then these feature values will be passed to a fully connected softmax layer. The output of a softmax layer is a probability distribution over labels [6][18].

For regularization one may apply dropout on the penultimate layer which sets a proportion p of the hidden units to zero in training step. More formally if $z = [\hat{c}_1, \dots, \hat{c}_m]$ assumed as penultimate layer, the output layer y, in forward propagation step, will be computed using

$$y = w \cdot (z \circ r) + b, \quad (12)$$

where $r \in \mathbb{R}^m$ is a vector containing Bernoulli random variables with parameter of p, and $\circ$ is an element wise multiplication operator. w and b are also the weight matrix and bias vector of the output layer (note that dropout is only used in training step) [6][18].

IV. Dataset and Experimental Result

A. Data

We collect a dataset containing a total of 200761 customer reviews about electronic products in Persian language using a web crawler. These reviews are gathered from the website www.digikala.com. We also include customer rates of each review, to label them as positive or negative. Since a few number of reviews had customer rate, a huge number of them remained without label. Detail statistics about dataset gathered, is reported in Table I. In This table (N_+, N_-, N_u) are respectively the number of positive, negative and unlabeled examples. l is the average number of tokens per examples and V, is the number of different token types in the corpus.

TABLE I. DATASET STATISTICS.

Dataset	Detailed Statistics			
	C	(N_+, N_-, N_u)	l	V
Digikala reviews	2	(45839,4196,150726)	59.73	187113

B. Training

Since there was no large corpus of supervised training data in Persian, we initialized word vectors with those obtained in an unsupervised way to improve the performance. Therefore we trained a skip-gram model with all of the documents in our corpus. After training word vectors we used them as input to the CNN and Bidirectional-LSTM models. Then these models were trained using labeled data.

We randomly select 90% of all labeled data as train set. The remaining of them used as test set. For both models we tried to minimize cross entropy loss function between the outputs of the softmax layer and their corresponding labels. This loss function is shown in (13),

$$loss(\hat{y}, y, w) = \frac{1}{N} cross\_entropy(\hat{y}, y) + \lambda \|w\|_2 \quad (13)$$

where $\hat{y}$ is the set of outputs of the softmax layer for training samples and y is the set of corresponding labels. N is the number of training samples. w also denotes the weight matrix at the penultimate layer in the models and λ is a coefficient which controls the importance of the regularization term.

In our setting we consider the dimensionality of 150 for word vectors. For training these vectors we used a window of size 5 in skip-gram model.

In supervised training the regularization coefficient was set to 0.1. We used Adam optimization algorithm with learning rate of 0.001 to minimize the loss function. Dropout rate (p) of 0.5 and batch size of 100 were also used. Note that the dropout was only used in train step.

For CNN model we used filters with height (h) 3, 4, 5. As mentioned before the width of these filters are equal to the dimension of word vectors which in our setting is 150. For each filter size there were 150 feature maps.

For Bidirectional-LSTM model the number of hidden units was set to 200.

C. Evaluation metrics

The dataset collected is unbalanced i.e. the number of positive and negative samples are unequal. Therefore we used precision, recall and F-score to evaluate the models. Because the positive samples were about 10 times more than negative samples, we calculate precision, recall and F-score base on negative samples. We also reported a Confusion Matrix for each model we test on our dataset. The overall form of the reported confusion matrix is shown in Table II.

Equations (14) to (16) show how we compute precision, recall and F-score based on the negative samples using the confusion matrix.

TABLE II. OVERALL FORM OF THE CONFUSION MATRIX.

	Selected as Negative	Selected as Positive
Negative	TN	FP
Positive	FN	TP

$$Precision = \frac{TN}{TN+FN} \quad (14)$$

$$Recall = \frac{TN}{TN+FP} \quad (15)$$

$$F - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (16)$$

D. Baseline

We compare our results with one of the best traditional approaches called NBSVM-bi which was 91.22% correct on sentiment classification of Large Movie Review Dataset (IMDB) [10]. IMDB dataset of movie reviews is one of the largest datasets for sentiment analysis which is in English and is publicly available.

E. Results

Table III reports the experimental results of each individual model. The confusion matrixes of the models are also reported in Tables IV, V and VI. Results show that NBSVM-bi gives the best precision but CNN gives the best F-score across the other models. As mentioned before F-score considers precision and recall in its calculation. Therefore it can be concluded that methods based on deep learning performed better and in our experiments CNN had the best performance among the other models that we used.

One reason of the low performance in classification of negative samples is unbalanced training data, but despite this issue methods based on deep learning did better which means better generalization on negative samples. Actually the unsupervised learning step, which was used to learn vector representations of words, is one of the main reasons in better generalization as it uses the semantic information from a large corpus of text data.

TABLE III. PRECISION, RECALL AND F-SCORE MEASURES OF THE MODELS ON PERSIAN DATA REVIEWS.

Approach	Precision[%]	Recall[%]	F-score[%]
NBSVM-bi	70.7	31.9	44.0
Bidirectional-LSTM	54.2	52.2	53.2
CNN	59.1	52.2	**55.4**

TABLE IV. CONFUSION MATRIX FOR NBSVM-BI MODEL .

NBSVM-bi	Selected as Negative	Selected as Positive
Negative	123 (31.9%)	262 (68.1%)
Positive	51 (1.1%)	4568 (98.9%)

TABLE V. CONFUSION MATRIX FOR BIDIRECTIONAL LSTM MODEL .

Bidirectional LSTM	Selected as Negative	Selected as Positive
Negative	201 (52.2%)	184 (47.8%)
Positive	170 (3.7%)	4449 (96.3%)

TABLE VI. CONFUSION MATRIX FOR BIDIRECTIONAL CNN MODEL .

CNN	Selected as Negative	Selected as Positive
Negative	201 (52.2%)	184 (47.8%)
Positive	139 (3%)	4480 (97%)

V. CONCLUSION

In this paper, we studied two deep neural network architectures and their results in classifying documents based on their sentiment polarity. From the results we concluded that, methods based on deep learning had better F-score than NBSVM. One reason could be the use of word vector representations which is done in an unsupervised way. This learning step addresses most of the difficulties arise from different writing styles in Persian, and since the learning is unsupervised it doesn't suffer from lack of the dataset. We also introduced a new Persian dataset for sentiment classification purpose.

REFERENCES

[1] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Foundations and trends in information retrieval,* vol. 2(1-2), pp. 1–135, January 2008.

[2] B. Liu, "Sentiment Analysis and Opinion Mining," *Synthesis lectures on human language technologies*, vol. 5(1), pp. 1–167, May 2012.

[3] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification," *Acl*, pp. 1555–1565, June 2014.

[4] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," *ICML*, vol. 14, pp. 1188–1196, June 2014.

[5] A. Bagheri, M. Saraee, and F. De Jong, "Sentiment Classification in Persian : Introducing a Mutual Information-based Method for Feature Selection." *21st Iranian Conference on Electrical Engineering (ICEE)*, pp. 1-6, May 2013

[6] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *arXiv preprint arXiv:1408.5882*, 2014.

[7] O. Irsoy and C. Cardie, "Opinion Mining with Deep Recurrent Neural Networks," *EMNLP*, pp. 720–728, 2014.

[8] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, vol. 10, pp. 79–86, July 2002.

[9] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions," *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 151-161, July 2011, Association for Computational Linguistics.

[10] S. Wang and C. D. Manning, "Baselines and Bigrams : Simple , Good Sentiment and Topic Classification," *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers*, vol. 2, pp. 90-94, July 2012, Association for Computational Linguistics..

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[12] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 142–150, June 2011, Association for Computational Linguistics.

[13] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic Compositionality through Recursive Matrix-Vector Spaces," *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1201-1211, July 2012, Association for Computational Linguistics.

[14] R. Socher, A. Perelygin, and J. Wu, "Recursive deep models for semantic compositionality over a sentiment treebank," *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631, pp. 1631–1642, October 2013.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *Advances in neural information processing systems*, pp. 3111-3119, 2013.

[16] P. Wang, Y. Qian, F. K. Soong, L. He, and H. Zhao, "A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding," *arXiv preprint arXiv:1511.00215*, 2015.

[17] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[18] Y. Zhang and B. C. Wallace, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification," *arXiv preprint arXiv:1510.03820,* 2015.