International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-9 Issue-1, May 2020

Sentiment Analysis using Multiple Word Embedding for Words

S. K. Shirgave, P. M. Gavali

Abstract: ***Nowadays users express their opinions on different websites like e-commerce and special review websites. Analyzing customers' opinions and their responses is important for decision making. So the researchers worked on analyzing these reviews automatically using a classical machine learning approach like Support Vector Machine (SVM) and various modern deep neural networks. For these networks, words are represented by using vectors called word embeddings. The required word embeddings are taken from pre-trained Word2Vec or learned from a corpus of the given main task. But, each method has its demerits. In the case of pre-trained word embeddings, embeddings are learned from large general corpus so these embeddings are not task specific. While in the case of learning words from the corpus of the main task, it does not reflect the true semantics. To deal with these problems, we have proposed an embedding developer model. This model develops task specific word embedding which also reflects true semantics. Task specific word embeddings are generated from the given corpus using the embedding layer in Keras. It builds the embeddings by considering relationships between words in the window. While true semantics are taken from Word2Vec embeddings. The proposed model combines these two embeddings to generate true semantics and task specific word embeddings. Result analysis shows that the proposed system works better on many benchmark datasets.***

Keywords **:** ***Deep Learning, Embedding, Pre-trained Model, Sentiment Analysis***

I. INTRODUCTION

In this digital era, a huge amount of information is available on the internet. This information needs to be analyzed for decision making. Natural language processing works on this information, to uncover useful hidden information. In this direction, Sentiment Analysis (SA), a subpart of natural language processing, works on analyzing the information for identifying the sentiment of the writer. It identifies whether the given piece of text carries positive, negative, or neutral sentiments. Identifying sentiments of given text manually is a difficult and time-consuming task. So, researchers identified different ways to identify the sentiment by using Bay's classifier [1], Support Vector Machine [2][3], the dictionary [4][5], and deep neural networks [6][7][8][9][10].

Deep neural networks provide accurate results than the other classical models [7][8][9]. These models take word embeddings [11] as input. Word embeddings represent words in semantic space. Many pre-trained models like Word2Vec provides word embedding. But these pre-trained models are trained on huge data. Such embeddings are not task-specific. Another way is to learn embeddings from the training data itself [29]. But most of the time available data is small and may not learn the embedding well. In this paper, we have developed two different word embeddings for each word using pre-trained and locally trained word embeddings for sentiment analysis. Result analysis shows that modified embedding provides accurate results compared to the pre-trained and locally learned embeddings independently.

The rest of the paper is organized as follows: In section II, we elaborated on the background of word embeddings and sentiment analysis models. In section III, we describe the motivational cases while in section IV we introduce a mathematical model. Section V defines the architecture of the system. In section VI, we discuss the implementation details with result analysis while in section VII we conclude the paper with future scope.

II. BACKGROUND

In this section, we describe the different methods used for building word embeddings to represent words and models used for identifying sentiment analysis using these word embeddings.

In natural language processing, sentiment analysis was identified by using a sentiment dictionary [4][12]. If sentiment words are present in the text then by using different natural language rules, sentiments of text were identified. These methods required handcrafted and language rules. To remove this problem, researchers introduced machine learning models. These models were: probability based Bay's Classifier [1][3], Entropy based classifier [1], and Support Vector Machine [13]. While working with these models, there was a need for a co-occurrence matrix and mapping of a word to a vector. The simplest way to form a vector for a word is by using 'one-hot encoding'. In this method, the size of the word vector is equal to the size of the vocabulary. All the vector positions are zero, except the word index in vocabulary is set to one. But these word vectors are bulky and sparse.

Manuscript received on April 02, 2020.
Revised Manuscript received on April 15, 2020.
Manuscript published on May 30, 2020.
* Correspondence Author
Dr. S. K. Shirgave*, Computer Science and Engineering, DKTE Society's Textile and Engineering Institute, Ichalkaranji, India. Email: skshirgave@gmail.com
Mr. P. M. Gavali, Computer Science and Engineering, DKTE Society's Textile and Engineering Institute, Ichalkaranji, India. Email: gavalipm87@gmail.com

It also does not represent any semantic meaning through this representation. So the researchers [14] used a co-occurrence matrix of words along with Pointwise Mutual Information (PMI) and single value decomposition to get comparatively small-sized, real-valued vector to represent the word. Other researchers [11] also came with two models namely continuous bag-of-words and skip-gram.

The continuous bag-of-words model predicts the next word for the given series of words while the skip gram model predicts surrounding words for given single context words. These models provided semantic oriented word embeddings by considering the context of the word. The Skip-gram model was trained on a large corpus of Wiki to provide word embedding. These pre-trained word embeddings are used by different deep learning models comprising of convolutional neural network [8], recurrent neural network [10], gated recurrent network [7], long short term memory networks [8] and their variations to the basic models. These models provided better results than classical machine learning approaches.

Since these word embeddings are generated over large general corpus, these embeddings are not task-specific. So many researchers [15] worked on modifying embeddings generated from the pre-trained model. Duyu [16] worked on developing sentiment oriented word embeddings for sentiment analysis. Also, researchers [17] have developed the word embeddings for words in different domains. There are some cases where the same word in different domains indicate opposite sentiment. In such cases, the developed domain knowledge aware word embeddings work better.

III. MOTIVATIONAL CASES

In this section, we consider two motivational cases: first is with Word2Vec, and the second is with local word embeddings learned from training data itself.

Word2Vec is effective execution of Continuous bag-words-vector and skip-gram model which provides dense and lower dimensional vectors called word embeddings. These word embeddings are learned from a huge corpus. Thus these word embeddings represent a word in semantic space. But these word embeddings are not task specific. As stated in the paper [15], in the case of sentiment analysis, the word embeddings for good and bad should be away from each other. But word embeddings for these words are close to each other. These embeddings need to be refined for a specific task like sentiment analysis.

Another way to generate the embeddings is to learn the embedding by considering the given corpus itself. But most of the time training data is less and embeddings do not carry true semantics between the words. For example, if in a movie review, the word 'good' occurs with 'movie' frequently then it will learn the embeddings and will indicate the word embeddings for 'good' and 'movie' close to each other. So if in review, we are getting the word 'movie' then it will mislead to positive. Such embeddings need to be refined.

In the first case, we need to consider the training data itself for building word embeddings so we get local context. In the local context, it is less likely that 'good' and 'bad' will occur in the same context or statement. So embeddings learned for 'good' and 'bad' will be away from each other. While in the second case we need to consider global context so 'movie' and 'good' should not be close to each other, unlike local context. Thus in the proposed system, we have used both these embeddings for identifying the sentiment.

IV. MATHEMATICAL MODELING

With the above motivational cases, in this section, we describe the mathematical model for building the new word embeddings and deep neural network.

Let E_{Local} be the local word embedding learned from the given text corpus. Let k be the number of context words (W) considered from the given text for generating the local embeddings. E_{Local} is learned by the following equation

$$E_{Local} = \varphi(W_1, W_2, \dots, W_k) \quad (1)$$

E_{Global} is the word embeddings learned from large global corpus provided by the Word2Vec. Then the new word embeddings E_{New} are identified by concatenating these embeddings as follows:

$$E_{New} = E_{Global} \,||\, E_{Local} \quad (2)$$

Output O_{output} for sentiment model can be given by,

$$O_{Output} = \sigma\,(M^T E_{New}^{Text} + b) \quad (3)$$

Since we are working with a binary sentiment classification problem, we have used the sigmoid activation function (σ) in the last layer of Deep Neural Networks. It takes transpose of weight matrix (M) for enabling matrix multiplication with the embedding matrix of the input text (E_{New}^{Text}) generate from E_{New}. In the equation, b is bias. The M and b are the learning parameters. These parameters will be learned by Deep Neural Network using an optimization algorithm.

V. ARCHITECTURE

In this section, we describe the model used for generating the modified word embeddings. Fig. 1 shows the architecture of the system.

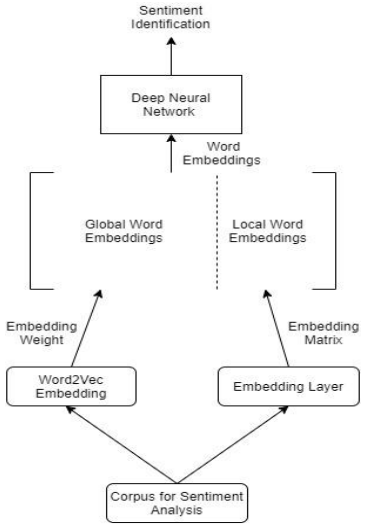

Fig. 1 Embedding Developer Model

The system takes a text corpus for sentiment analysis. After lemmatization, removing punctuation, and spell check; vocabulary for a given corpus is created. For each word in the vocabulary, the corresponding word embedding from Word2Vec is identified. Collection of all these word embeddings forms embeddings weight matrix. These embeddings are called 'Global Word Embeddings'. The same vocabulary words are used by Embedding Layer for building local word embeddings. Embedding Layer learns the word embeddings for the words from the given corpus only. It generates an embedding matrix. These embeddings are called 'Local Word Embeddings'. Now for each word, there is local as well as global word embedding. These word embeddings are combined to form new word embeddings. Such word embeddings are taken as first layer weight by the deep neural network. This deep neural network identifies sentiments.

VI. IMPLEMENTATION AND RESULT ANALYSIS

In this section, we discuss the implementation details and effectiveness of newly introduced word embeddings with different models of sentiment analysis. While implementation, we have used following algorithm for developing new embeddings.

Algorithm: New Embedding Builder
Input: Corpus, Word2Vec
Output: New Embeddings
Steps:
1. Split text with whitespace to identify word
2. **foreach** word **in** corpus **do:**
   if word is NOT alphabets **then** remove word
   if word is stop word **then** remove word
   if word is NOT base word **then** convert to base word
   if word is punctuation **then** remove word
   if length(word) <=1 **then** remove word
   word <- stemming(word)
   vocabulary <- add(word)
3. Assign integer number to each word from vocabulary
4. Convert corpus to sequence of integer
5. model <- load pretrained Word2Vec model
6. global_ embedding <- model(word)
7. local_ embedding <- Embedding Layer (largest_assigned_integer, embedding_dimension, max_length)
8. new_embeddings <- concatenate (global_ embedding, local_embedding)

The first step in the proposed system is preprocessing the given corpus. We have generated the tokens and removed stop words from the corpus. A list of stop words is taken from the NLTK library. We have used potter stemming for stemming the words. Besides this, we also checked whether the available tokens contain only alphabets or not. If not, we have removed those token.

For implementing the proposed model, we have used Keras with Tensorflow 1.X as backend. To generate local embeddings from the given text, the whole corpus needs to be converted in the sequence of numbers. For this, we have used the Tokenizer class. The method fit_on_text provides the unique mapping between the word and integer. With this mapping, the method text_to_sequence converts the corpus into a sequence of the integer. In Keras, the embedding layer learns local word embeddings. This is the first layer of the model. It takes a sequence of integers generated previously which represent the given text. The layer considers these integer numbers in windows and learns the relationship between the word of the same window. After training the network over the given corpus, we get weight for the embedding layer. These embeddings are generated by the automatic differentiation engine of Tensorflow. In this embedding weight matrix, row number corresponds to the integer assigned to the word. So internally, each row represents a word embedding for word. Thus we get local embedding for each word. For generating these local embeddings, we have to provide vocabulary size and word embedding size. The size of the word embedding is a hyperparameter. While to get global embeddings, we have used Google Word2Vec pre-trained model. Gensim library loads the Word2Vec model in the key vector format. To get word embedding for a specific word, we have to pass the word as an input to the loaded Word2Vec model. It provides embeddings for that word. After getting these local and global word embeddings, new embeddings are formed by combining these two embeddings together. All these three word embeddings are tested over the following dataset.

IMDB [31]: IMDB is a famous movie review dataset for sentiment analysis. It consists of 25000 training and 25000 testing reviews.

Dataset	Count	Mean	Std	Min	Max
IMDB	50000	0.5	0.5	0	1
Amazon Fine Food	525814	4.27	1.31	1	5
Electronic Gadgets	5649	0.66	0.47	0	1
Twitter	1599999	0.5	0.5	0	1

Table 1 Statistical Measures of different datasets

Amazon Fine Food Review [32]: It includes more than half a million food reviews. These reviews are star rated. Reviews with one and two stars are treated as negative while reviews with four and five are treated as positive reviews.

Electronic Gadgets Reviews [33]: This dataset consists of reviews written by different users for various electronic gadgets like mobile, and laptop. The dataset includes subjective and objective reviews. For sentiment analysis only subjective reviews are important. So from the dataset, we have only considered subjective reviews.

Twitter Dataset [20]: This dataset consists of more than 1.5 million twits extracted from Twitter for sentiment analysis.

Table 1 indicates the size of the dataset, mean, standard deviation, minimum, and maximum values for different datasets. It shows that IMDB and Twitter datasets are evenly distributed in sentiment classes while amazon fine food and electronic gadgets datasets slightly deviate.

We have developed deep learning models for identifying sentiments of a given text. For comparing the results with different embeddings we have used accuracy metrics. It is a ratio of the number of correct predictions to the total number of predictions. So it indicates a fraction of correct predictions our model got right. Same models are used with Global, Local Embeddings, and New Embeddings.

To start with some random weights, we have used 'Lecun Uniform Initializer'. It assures that the same random weights will be generated at each time. So, while using Global, Local, and New embeddings, we start from the same location on the error surface. Table 2 shows the accuracy of deep learning models with global, local, and new word embeddings on different datasets. We have used 2 epochs for each model.

Dataset	Global Embedding	Local Embedding	New Embedding
IMDB	61.86	87.35	**87.60**
Amazon Fine Food Review	84.21	94.99	**99.52**
Electronic gadgets Review	64.40	64.40	**64.40**
Twitter	65.79	70.94	**73.61**

Table 2 Accuracy of Deep Learning Model with different Embeddings on various Benchmark Dataset

From table 2, it is observed that local embedding provides a better result than Word2Vec. Using these local and global word embeddings together, the result even improves on various benchmark datasets.

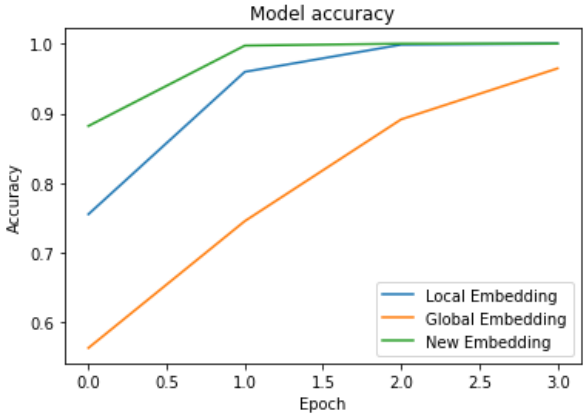

Fig. 2 Model Accuracy on IMDB Dataset with Global, Local and New embeddings

Fig. 2 shows the accuracy of global, local, and new embeddings for the IMDB dataset. We have taken four epochs for the experiment. The graph clearly shows that the performance of local and new embedding is better than global embedding. While comparing local and new embedding, it is observed that new embedding converges quickly than local embedding. After some epochs, both lines are close to maximum accuracy.

VII. CONCLUSION AND FUTURE SCOPE

Natural language processing analyzes the text for taking different decisions. Sentiment Analysis also works in the same direction to identify the sentiment of a given text. Many models identified sentiments of text. These models work on word vector called word embeddings. These embeddings are obtained from the pre-trained model or developed from local training data. But both these methods have their problems. Pre-trained word embeddings are trained on large data but it is not task specific while local word embeddings learned from training data but may not reflect the true semantics. So we have created a new model that develops new word embedding by using pre-trained as well as local word embedding. Result analysis shows that with modified word embeddings we are getting better results. In summary, each word carries global and local meaning in the text. Local meaning completely depends on the context in which word is used while global meaning is drawn from the general corpus. In case of sentiment analysis, we have to consider both these global and local meanings. Consideration of both global and local contexts provides a better result than considering each individually.

In this paper, we have worked on identifying new word embeddings from global and local word embedding. While generating local word embeddings, we have used a given domain specific corpus. For each domain, we have to learn these local embeddings. In the future, we can modify these local word embeddings with a different domain. Further, we can modify embeddings by assigning the learnable sentiment weight to each word to decide sentiment orientation of word.

REFERENCES

1. B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques", in Proc. Assoc. Comput. Linguistics Conf. Empirical Methods Natural Lang. Process., vol. 10, pp. 79-86, 2002
2. B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts", in Proc. 42nd Annu. Meeting Assoc. Comput. Linguistics, pp. 115–124., 2004.
3. 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 of the ACL, pg. 115–124, June 2005.
4. M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis", Association for Computational Linguistics, Vol.37, No.2, 2014.
5. G. Zhao, X. Qian, and X. Xie, "User-service rating prediction by exploring social users' rating behaviors", IEEE Transactions on Multimedia, vol. 18, No. 03, March 2016..
6. R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. 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, Oct. 2013
7. D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification", in Proceedings of the 2015 Conference on Empirical Methods in natural Language Processing, pg. 1422–1432, September 2015
8. C. Zhou, C. Sun, Z. Liu and F. Lau, "A C-LSTM neural network for text classification", arXiv, 2015.
9. D. Tang, B. Qin, F. Wei, L. Dong, T. Liu, and M. Zhou, "A joint segmentation and classification framework for sentence level sentiment classification", IEEE/ACM Transactions on Audio, Speech, and Language Processing, Vol. 23, No.11, Nov. 2015
10. P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning", in Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pg.2873-2875,2016.
11. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality", in Proc. Conf. Neural Inf. Process. Syst., pp. 3111–3119, 2013.
12. Xiaojiang Lei and Xueming Qian, "Rating Prediction via Exploring Service Reputation", IEEE 17th International Workshop on Multimedia Signal Processing (MMSP), Oct 19-21, 2015
13. Nurulhuda Zainuddin and Ali Selamat, "Sentiment Analysis Using Support Vector Machine", IEEE 2014 International Conference on Computer, Communication, and Control Technology (I4CT 2014), September 2 - 4, 2014 - Langkawi, Kedah, Malaysia

14. Aleksander Wawer, "Mining Co-Occurrence Matrices for SO-PMI Paradigm Word Candidates", Proceedings of the EACL 2012 Student Research Workshop, pages 74–80, Avignon, France, 26 April 2012.
15. L. Yu, K. Lai, and X. Zang , "Refining Word Embeddings Using Intensity Scores for Sentiment Analysis", IEEE/ACM Transactions On Audio, Speech, And Language Processing, Vol. 26, No. 3, March 2018
16. Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Lue and Ming Zhou , "Sentiment Embeddings with Applications to Sentiment Analysis", IEEE Transactions On Knowledge And Data Engineering, Vol. 28, No. 2, February 2016
17. Auro Dragoni, and Giulio Petrucci, "A neural word embeddings approach for Multi-domain sentiment analysis", IEEE transaction on Affective Computing.
18. Z. Yangsen, J. Yuru, and T. Yixuan, "Study of sentiment classification for Chinese microblog based on recurrent neural network", Chinese Journal of Electronics, vol. 25, no.4, July 2016.
19. D. Tang, F. Wei, B. Qin, M. Zhou, and T. Liu, "Building large-scale twitter-specific sentiment lexicon: A representation learning approach," in Proc. 25th Int. Conf. Comput. Linguistics, 2014, pp. 172–182.
20. D. Tang, F. Wei, B. Qin, T. Liu, and M. Zhou, "Coooolll: A deep learning system for twitter sentiment classification," in Proc. 8th Int. Workshop Semantic Eval., 2014, pp. 208–212
21. J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in Proc. Conf. Empirical Methods Natural Lang. Process., 2014, pp. 1532–1543.
22. R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in Proc. Annu. Meeting Assoc Comput. Linguistics, 2013, pp. 455–465
23. T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in Proc. Conf. Empirical Methods Natural Lang. Process., 2005, pp. 347–354.
24. A. Severyn and A. Moschitti, "On the automatic learning of sentiment lexicons," in Proc. Conf. North Amer. Ch. Assoc. Comput. Linguistics, 2015, pp. 1397–1402
25. P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J.C. Lai, "Class-based N-gram models of natural language," Comput. Linguistics, vol. 18, no. 4, pp. 467–479, 1992.
26. E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in Proc. 50th Annu. Meeting Assoc. Comput. Linguistics: Long Papers, 2012, pp. 873–882.
27. S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," J. Artif. Intell. Res., vol. 50, pp. 723–762, 2014.
28. A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in Proc. 7th Int. Conf. Lang. Resources Evaluation, vol. 2010, 2010.
29. Keras documentation: https://www.keras.io
30. Xiaojiang Lei, Xueming Qian and Guoshuai Zhao, "Rating Prediction Based on Social Sentiment From Textual Reviews", in IEEE Transaction on Multimedia, Vol. 18, No. 9 , September 2016
31. Maas, Andrew L., Daly, Raymond E., Pham, Peter T., Huang, Dan, Ng Andrew Y., Potts, Christopher, " Learning Word Vectors for Sentiment Analysis", Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, June 2011
32. Julian McAuley and Jure Leskovec, "From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews", International World Wide Web Conference Committee (IW3C2), 2013
33. Guan Z, Chen L, Zhao W, et al., "Weakly-supervised deep learning for customer review sentiment classification", Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. AAAI Press, 2016: 3719-3725.

AUTHORS PROFILE

Dr. S. K. Shirgave, is currently working as an Associate Professor in the Department of Computer Science and Engineering at DKTE Society's Textile and Engineering Institute, Ichalkaranji. He has completed a Ph.D. in Web Page Recommender System. He has published many papers in a national and international journal. He also presented many papers in conference papers. His area of interest is Web Mining, Security, and Recommender System.

Mr. P. M. Gavali, is a Research Scholar in Computer Science and Engineering from Shivaji University, Kolhapur. He has completed his M. Tech. in Computer Science and Technology from Shivaji University Kolhapur. His area of interest is Natural Language Processing and Deep Learning. He is currently working as an Assistant Professor in the Department of Computer Science and Engineering at DKTE Society's Textile and Engineering Institute, Ichalkaranji.