HORA 2019
www.horacongress.com

International Congress on Human-Computer Interaction, Optimization and Robotic Applications

July 5-7, 2019, Urgup, Nevşehir, Turkey

SETSCI Conference Proceedings
www.set-science.com
8: 106 - 111, 2019

Word Vector Space for Text Classification and Prediction According to Author

İlknur Dönmez[1*], Elnaz Pashaei[2] and Elham Pashaei[3]

[1]*Computer Engineering, İstanbul Aydın University, İstanbul, Turkey*
[2]*Software Engineering, İstanbul Aydın University, İstanbul, Turkey*
[3]*Computer Engineering, İstanbul Gelişim University, İstanbul, Turkey*
[*]*(ilknurdonmez@aydin.edu.tr)*

Abstract – Word embedings are intense vector representations of words. They capture the semantic order between words. In our study, we propose a new method for classifying text by author, using word embeds and author vector space. Using each author's books, we trained the data to create author-specific word vector space. Word vector spaces consist of vector representations of all the words used by the author in the training dataset. In our study, different word vector spaces were created for different authors. The aim here is answering the question "Which author is more likely to write any given text?". In a text, we propose a method to find out whether consecutive word vectors belong to that author's vector space. Our method is based on a basic principle used when constructing word embedding vectors. Extensive results on the data sets show that the proposed method performs better than the latest technology methods in terms of accuracy.

Keywords – word embeding, word vectors, Turkish, text classification according to author

I. Introduction

Natural language processing (NLP) is an important artificial intelligence technology in understanding complex human language. Word vectors are one of the most advanced researches on deep learning, performing well in many different NLP fields, including question answering and machine translation [1].

In traditional natural language processing systems, words are seen as separate atomic symbols. Each word is expressed with a random index value or a binary representation (one-hot-vector). These encodings are random and do not contain any useful information about the relationships between words. To represent each word as unique and discrete increases data sparsity. Therefore, if words are represented in this way, more data will be needed to train statistical models. Using vector representations can overcome some of these obstacles.

Vector space models (VSMs) [2] represent (embed) words in continuous vector space where semantically similar words are close to each other in this space. The method is based on the Distribution Hypothesis, which states that the words used in the same context are semantically close to each other. Word2vec is a predictive model for learning word embeddings from raw text.

Although there are many English word2vec studies in the literature, there exist a small number of Turkish word2vec applications. In English, word2vec are used in Sentiment analysis [5], [6], document and text similarity [7], text classification [8], [9], [10] and author profiling [11], [12], [13], [14]. For Turkish, word embedding's were used in Document classification [15], sentiment classification [16], tweet sentiment analysis [17], [18] and event detection [19].

There is also network-based Predictive Text Embedding method which is used for classification. In this method, the word to word networks, word to document, and word to label network is generated. It uses word2vec methods to find the value of each edge [20].

For sentence vector representation task there are some studies. It is first proposed in 2017. It uses each word vector which is created using word2vec. Generating Sentence Embedding has two steps. First step is encoding a sentence in a linear weighted combination of the word vectors and the second step is performing a common component removal (removing the projection of the vectors on their first principal component) [21]

As a result of the literature review, it is seen that there is no text classification according to the author on Turkish documents. In a text classification problem, the task is assigning each document or text to exactly one author. In our study, Oğuz Atay who is one of the master names of Turkish Literature and Nasrettin Hoca who is legendary with his unique style and jokes are chosen for classification problem. Wor2vecs were trained with the book texts of the authors and by using these word2vecs, our model estimates, which paragraph belongs to which author?

Also for solving the classical classification problem, the contribution of word vector spaces to the classification problem is shown. To determine whether a data belongs to a particular field, word vector spaces generated using field-specific databases can be used.

Section II describes the discovery of data and word vector spaces and how these spaces contain similarity. Section III describes the basic principle, hypothesis, and the model we used for the classification problem. Section IV is the section where our results are compared with those of traditional

DOI: 10.36287/setsci.4.5.021

algorithms. Chapter V discusses the method and its uses. Section VI is the conclusion section.

II. DATA AND METHOD

In this section, we will describe the data and methods used when conducting the study.

A. Data

Word vector space for two different authors was created using the data in Table.1. The source texts for each author are divided into two parts. The first was used to train and construct embedding vectors. The other part is reserved for testing purposes.

As shown in Table 1, we used 6215 sentences for the training of word vector space produced from Nasrettin Hoca texts. For this author, 500 sentences were used in the test stage. The number of vectors in the vector space obtained after the training is 8,144. It also equals the number of vocabulary for Nasrettin Hoca.

Table 1. Data Summary for Nasrettin Hoca

Train Data related with Nasrettin Hoca	
Character Number	1.286.322
Word and punctuations Number	181.179
Sentence Number	6.215
Vocabulary Length	8.145
Test Data related with Nasrettin Hoca	
Character Number	176.346
Word and punctuations Number	11.913
Sentence Number	500
Vocabulary Length	645

As seen in Table.2, for word embedding's that are generated from Oğuz Atay texts, we used 35260 sentences. For testing, 1000 sentences were used. After the training word embedding's number are 15.624 which means there is 15.624 word vectors in the word vector space.

Table 2. Data Summary for Oğuz Atay

Train Data related with Oğuz Atay	
Character Number	2.411.329
Word and punctuations Number	321.968
Sentence Number	35.260
Vocabulary Length	15.624
Test Data related with Oğuz Atay	
Character Number	800.012
Word and punctuations Number	122.169
Sentence Number	1000
Vocabulary Length	4.816

B. Word Embedding Creation

In our study, to represent the words with a vector, word embedding was created. These vectors contain some semantic relationships between words. One way to create such embeds is the word2vec method, as suggested by Mikolov. In this method, basically, the words used in the training set are initially represented by a vector of random numbers (One-hot vector). During the training phase, the vector representation of the word is learned by guessing with adjacent words.

The very large corpus is used as training data. There are two methods to get Word2vec, Continuous Bag of Word model (CBOW) and Skip-Gram model. CBOW predicts target words from source context words [22], while Skip-Gram predicts source context words from target words [23]. To find the vector representation of a particular word, the number of words used before and after that word is called a window size.

In our study, the texts of two different authors were used as an input to the word2vec application of Gensim [24]. The window size used was 5.

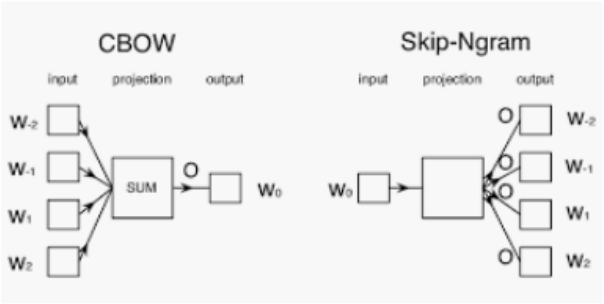

Fig.1 CBOW and Skip-NGram Architecture

C. Vector Space for Two Authors

In Figure 2 the word vector space for Narettion Hoca is seen. The vector dimension is 2. In the figure these dimentions are x and y. Each dot represents a word. Nasrettin Hoca's texts are trained using word2vec to generate this word vector space.

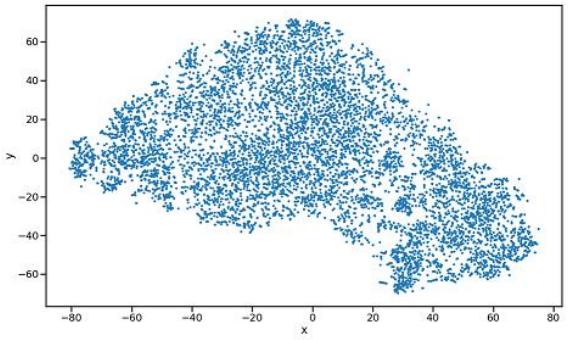

Fig. 2 Nasrettin Hoca Vector Space

In Figure 3 the word vector space for Oğuz Atay is seen. The vector dimension is 2. Oğuz Atay's texts are trained using word2vec to generate the word vector space specific to Oğuz Atay.

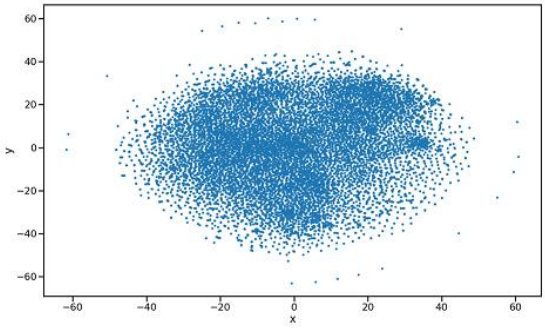

Fig. 3 Oğuz Atay Vector Space

There's an important point here. The two authors are likely to have common words in their dictionaries. If we look at the vector value of a common word in two spaces, this vector value differs in the vector space of Author A and in the vector space of Author B, because word vectors are constructed by looking at their neighbours in a large corpus. Also, this large corpus is different for each author.

D. Finding Similar Word Using Word Embeding Space

In this section, we have tried to show whether the word embedding vectors represent the words semantically in a correct manner. Finding a semantic relationship between words in natural language processing is one of the most challenging issues. It is seen that the word vectors give very successful results.

The most similar words to "Altın=gold" is seen in Fig.4. Here "Altın" is the searched word and we used three auxiliary words "Mal", "Gümüş" and "Para" to find the other similar words in Nasrettin Hoca word embedding space. As seen on the figure the automatically suggested similar words are "sekiz=eight", "dokuz=nine", "On=ten", "yüz=one hundred", "doksan=ninety", "Altınları= his gold" and "akçe", "dinar", "kuruş" which are the Ottoman money units. Because Nasrettin Hoca lived during Ottoman Empire, these currencies were used instead of gold at that time.

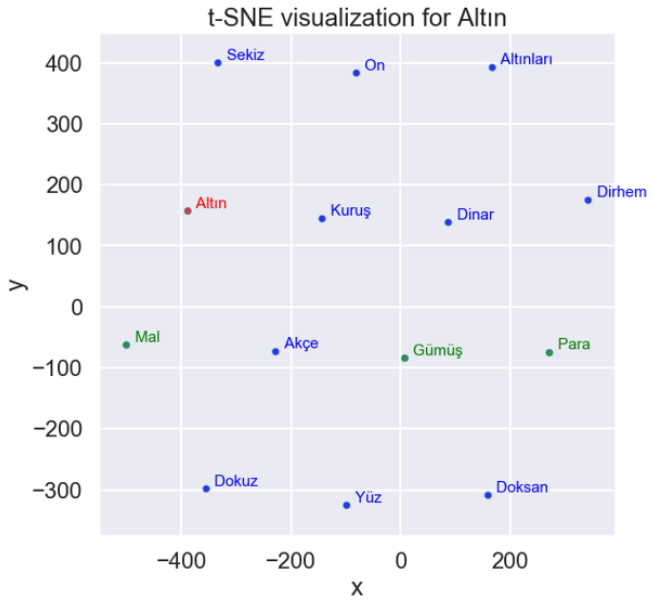

Fig. 4 t-SNE visualization for "Altın"

The 10th most similar words to "Eşek" are ploted with blue and the next 10th most similar words to "Eşek" are ploted with green as seen on Fig.5 in Nasrettin Hoca word embedding space. It is a zoomed field for only 21 items in vector space on Fig.2. Here "eşek" means "donkey" in English. As seen on the figure the other similar words are "at=horse", "eşeğe=to donkey", "atla=with horse", "donkey's=eşeğin" and "donkeys'=eşeklerin" which are really very similar in meaning.

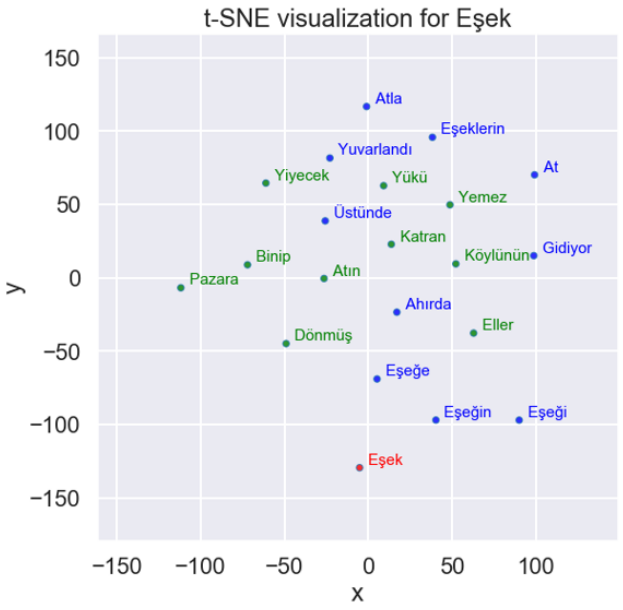

Fig.5 t-SNE visualization for "Eşek"

The most similar words to "Masa" word are plotted with blue and the auxiliary word "sandalye" is plotted with green as seen on Fig.6 in Oğuz Atay word embedding space. Here masa means "table" and sandalye means "chair" in English.

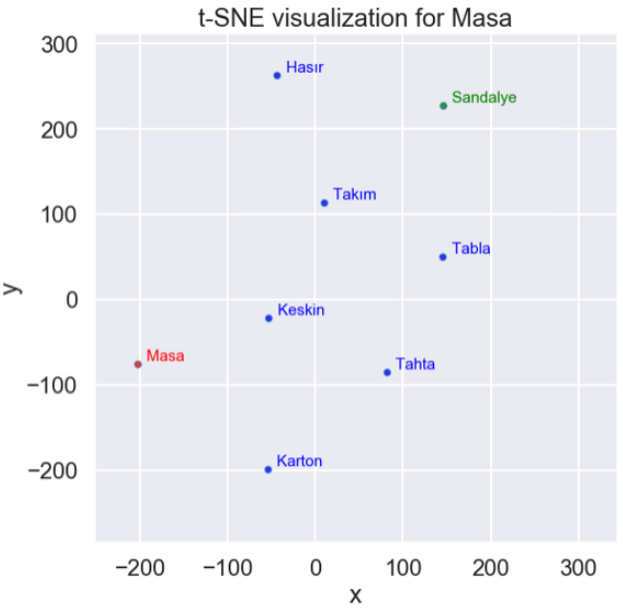

Fig.65 t-SNE visualization for "Masa"

E. Methodology

Preliminary principle: Because a word vector value is determined according to its neighbours vector value. The neighbour words vector values will be similar according to the training set.

"Vertices with similar probability distributions over the "contexts" are assumed to be similar"

As proposed by Mikolov [1], while finding word embedding's, the formula (1) is used. For a given window size T, for each center word we try to maximize the value of seeing Wc+t after Wc or we try to minimize the negative version as seen on equation (1).

$$J(Q) = -\frac{1}{T}\sum_{n=1}^{T}\left(\sum_{-m<j<m}\log(\mathrm{p}(Wc+t\,/\,Wc))\right) \quad (1)$$

In formula (1), $\mathrm{p}(Wc+t\,/\,Wc)$ is the probability function. Wc denote the center word; Wn denote the neighbour word of center word. It is about while Wt is seen what is the probability of seeing the neighbour words Wt+1, W-1 etc.

This probability is calculated with the formula (2).

$$\mathrm{p(Vn/Vc)} = \frac{\exp(Vn^{T}.Vc)}{\sum_{w=1}^{voc}\exp(Vn^{T}.Vc)} \quad (2)$$

In formula (2), the numerator is the point product of the neighbor word and the central word. This value indicates the similarity between the central word and its context. The denominator is used for normalization. The denominator is calculated by adding the similarity of the center word to all of the words in the dictionary. To calculate word vectors, a large text is given to the input, and the text is shifted one word at each iteration. The objective is to minimize formula (1). Assuming that we have determined the window size from the beginning, the updated variables are Vn and Vc. At the end of the Word2vec minimization process, word vectors for all words are generated.

Hypothesis: Authors have their own vocabulary. The authors also tend to use certain words consecutively. If an article is a work of the author, the words in the sentence must be in the word space used by the author. Also, since the sentence will be compatible with the consecutive word usage of the author, the total distance between consecutive word vector values will be small and the similarity will be high according to the rule in the preliminary principle -1.

Assume the sentence is generated from 1 to n words, the average similarity is calculated with a simple method.

$$\mathrm{S} = \{\mathrm{W_1}, \mathrm{W_2}, \mathrm{W_3}, \ldots\ldots, \mathrm{W_{n-1}}, \mathrm{W_n},\}$$

$$\mathrm{AvgSim} = \frac{1}{n}\sum_{i=1}^{n-1}\mathrm{CosinSimilarity(Wi, Wi+1)} \quad (3)$$

If A and *B are two vectors*, the cosin similarity cos (θ), is represented using a dot product and magnitude as:

$$\mathrm{Cos(Q)} = \frac{A.B}{||A||||B||} \quad (4)$$

To understand whether a sentence is close to the author's vector space, the sentence is first divided into words. Each word has values in all author spaces. For each author, the vector value of the word will be different.

In order to determine whether the sentence belongs to that author, the value of the words in that author's vector space is compared according to their similarities as consecutive pairs. Similarities between words in pairs are summed to find total similarity.

When the average similarity is calculated for each author, the sentence is said to belong to the author with the greatest similarity value.

If a word in the sentence is not in the author vector space, we skip that word vector, but since we still divide the total similarity by n, the average similarity will decrease. The fact that a sentence already contains a word that the author does not use in its large sources should reduce the likelihood that the sentence containing that word belongs to the author.

In our proposed method, we will decide that the sentence belongs to the author according to the mean value of similarity. We see average similarity values for each author; the maximum value points the author of the text.

- If (AvgSim_Author1) > (AvgSim_Author2); the text belongs to first author;
- If (AvgSim_Author2) > (AvgSim_Author1); the text belongs to second author;

III. Results

We compare our results with the classical text classification methods. In conventional methods, the training data and test data is used to process the classification model. In Table 3, our result is compared with the classical approaches. P is vector dimension number, K is class number.

Table 3. Complexity

Model	**Param**	**Complexity**	**Seq. Operation**
CNN	m.h.P	O(m.h.L.P)	1
LSTM	4.h.(h+P)	$O(Lh^2+hLP)$	L
SWEM	0	O(LP)	1
Our Model	K	O(KLP)	1

We compare our own model with CNN, LSTM, Simple Word Embedded Models (SWEM) [25]. For CNN, the size was taken as "m" for all filters. "h" represents the size of hidden layers in the LSTM or the number of filters in the CNN. P refers to the vector dimension.

Table 2 shows the number of parameters used in each model, the complexity of the calculation, and the sequential steps [25], [26]. Both CNN and LSTM have multiple parameters. Since K << m; h, the number of parameters in our models is less than CNN and LSTM models. In terms of computational complexity, our model is almost the same as the simplest SWEM model. Our model is smaller than CNN or LSTM by the factor mh / K or h / K.

Table 4: Test Accuracy on long text classification tasks, in percentage.

Model	Nasrettin Hoca	Oğuz Ayay
Bag of words	80.03	78.22
CNN	84.24	84.07
LSTM	85.53	86.01
SWEM	89.56	87.44
Our Model	93.63	90.27

Table 5: Test Accuracy on short sentence classification tasks, in percentage.

Model	Nasrettin Hoca	Oğuz Ayay
Bag of words	50.42	48.34
CNN	52.04	49.09
LSTM	56.12	53.56
SWEM	65.10	62.11
Our Model	82.10	78.23

As seen in Table 4 and Table 5, our model is compared with the classification algorithms of classical models. Traditional methods are successful in the use of large paragraphs but do not sufficiently succeed in the use of short sentences. Our model was compared with bag-of-word, CNN, LSTM, and SWEM methods. In short sentence classification, our model was found to be very successful compared to other methods.

IV. DISCUSSION

In this study, the vector space produced from the author's texts is used to determine whether any given text belongs to the author. This is a new approach not encountered in other studies. Let the words of a text be represented by vectors in the author's vector space. If the text belongs to the author, the distance between these consecutive word vectors is expected to be small. The reason for this is that when calculating these vectors (word2vec application), consecutive words are scored close to each other and eventually scored similarly. Our test results support this assumption. Using our method, it has been found that great accuracy values have been achieved especially in the classification of short texts. Other methods look especially at word frequencies. Since conventional methods look at the word frequencies or n-gram frequencies in the document to classify, they cannot achieve enough success in short texts. But with our approach, the author's vocabulary and habit of consecutive use of words are considered.

V. CONCLUSION

Our article aims to find out which text belongs to which author. For this purpose, the vector space of the author, which contains the words used by the author as a vector, is used. In this vector space, the vector values of consecutive words in the education set are close. Taking advantage of this feature, we have proposed a new method to find out if a text belongs to a particular author. Let us express consecutive words in a text with vectors in this author's vector space. This article suggests: The mean of the similarity between consecutive word vectors is a parameter. This allows you to determine if the text belongs to the author.

If the average of similarity is high, the text is likely to belong to the author. This new method yielded an accuracy increase of 3-4% compared to the old methods in texts consisting of 4-5 words in 2-classification.

In the classification of single sentences, a 16% increase in accuracy was observed. The proposed method can be used to create field-specific vector gaps if sufficient data is available, and to determine whether any information for that field belongs to that field. We believe that this method can be used in various fields such as bioinformatics.

REFERENCES

[1] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[2] Erk, K. (2012). Vector space models of word meaning and phrase meaning: A survey. Language and Linguistics Compass, 6(10), 635-653.

[3] Zhou, G., He, T., Zhao, J., & Hu, P. (2015). Learning continuous word embedding with metadata for question retrieval in community question answering. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (Vol. 1, pp. 250-259).

[4] Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722.

[5] Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 1555-1565).

[6] Dos Santos, C., & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers (pp. 69-78).

[7] Kusner, M., Sun, Y., Kolkin, N., & Weinberger, K. (2015, June). From word embedding's to document distances. In International Conference on Machine Learning (pp. 957-966).

[8] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in neural information processing systems (pp. 649-657).

[9] Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 1555-1565).

[10] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

[11] Rangel, F., Rosso, P., Potthast, M., & Stein, B. (2017). Overview of the 5-th author-profiling task at pan 2017: Gender and language variety identification in twitter. Working Notes Papers of the CLEF.

[12] Amir, S., Wallace, B. C., Lyu, H., & Silva, P. C. M. J. (2016). Modelling context with user embeddings for sarcasm detection in social media. arXiv preprint arXiv:1607.00976.

[13] Bayot, R. K., & Gonçalves, T. (2016, September). Author Profiling using SVMs and Word Embedding Averages. In CLEF (Working Notes) (pp. 815-823).

[14] Flekova, L., & Gurevych, I. (2015). Personality profiling of fictional characters using sense-level links between lexical resources. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 1805-1816).

[15] ŞahİN, G. (2017, May). Turkish document classification based on Word2Vec and SVM classifier. In 2017 25th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

[16] Uysal, A. K., & Murphey, Y. L. (2017, August). Sentiment classification: Feature selection based approaches versus deep learning. In 2017 IEEE International Conference on Computer and Information Technology (CIT) (pp. 23-30). IEEE.

[17] Ayata, D., Saraçlar, M., & Özgür, A. (2017, May). Turkish tweet sentiment analysis with word embedding and machine learning. In 2017 25th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

[18] Sarı, M., & Özbayoğlu, A. M. (2018, September). Classification of Turkish Documents Using Paragraph Vector. In 2018 International Conference on Artificial Intelligence and Data Processing (IDAP) (pp. 1-5). IEEE.

[19] Ertugrul, A. M., Velioglu, B., & Karagoz, P. (2017, June). Word embedding based event detection on social media. In International Conference on Hybrid Artificial Intelligence Systems (pp. 3-14). Springer, Cham.

[20] Tang, J., Qu, M., & Mei, Q. (2015, August). Pte: Predictive text embedding through large-scale heterogeneous text networks. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1165-1174). ACM.

[21] Yang, Z., Zhu, C., & Chen, W. (2018). Zero-training Sentence Embedding via Orthogonal Basis. arXiv preprint arXiv:1810.00438.

[22] Zhou, G., He, T., Zhao, J., & Hu, P. (2015, July). Learning continuous word embedding with metadata for question retrieval in community question answering. In Proceedings of the 53rd Annual Meeting

of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (pp. 250-259).

[23] Guthrie, D., Allison, B., Liu, W., Guthrie, L., & Wilks, Y. (2006, May). A closer look at skip-gram modelling. In LREC (pp. 1222-1225).

[24] Rehurek, R., & Sojka, P. (2011). Gensim—statistical semantics in python. statistical semantics; gensim; Python; LDA; SVD.

[25] Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., ... & Carin, L. (2018). Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. arXiv preprint arXiv:1805.09843.

[26]Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).