Int. J. Nonlinear Anal. Appl. 10 (2019) No. 2, 87-96
ISSN: 2008-6822 (electronic)
https://dx.doi.org/10.22075/ijnaa.2019.4177

A New Method for Sentence Vector Normalization Using Word2vec

Mohamad Abdolahi*, Morteza Zahedi

Kharazmi International Campus Shahrood University of Technology, Shahrood, Iran

(Communicated by J. Vahidi)

Abstract

Word embeddings (WE) have received much attention recently as word to numeric vectors architecture for all text processing approaches and has been a great asset for a large variety of NLP tasks. Most of text processing task tried to convert text components like sentences to numeric matrix to apply their processing algorithms. But the most important problems in all word vector-based text processing approaches are different sentences size and as a result, different dimension of sentences matrices. In this paper, we suggest an efficient but simple statistical method to convert text sentences into equal dimension and normalized matrices Proposed method aims to combines three most efficient methods (averaging based, most likely n-grams, and words mover distance) to use their advantages and reduce their constraints. The unique size resulting matrix does not depend on language, Subject and scope of the text and words semantic concepts. Our results demonstrate that normalized matrices capture complementary aspects of most text processing tasks such as coherence evaluation, text summarization, text classification, automatic essay scoring, and question answering.

Keywords: Text Preprocessing, Sentence Normalization, Word Embedding, Word Vector, Sentence Vector.
2010 MSC: 68T50

1. Introduction

Representing textual sequences such as words is a fundamental component of all text processing systems. Some of previous works uses words as the smallest units in man-made or machine generated

*Mohamad Abdolahi
Email addresses: mabdolah512@yahoo.com (Mohamad Abdolahi*), Zahedi@shahroodut.ac.ir (Morteza Zahedi)

Received: December 2017 *Revised:* November 2018

documents [1] [2]. There are some other studies which have used character-based models and convert sequences of characters into vectors [3] [4] [5]. But most researchers have been carried out on sentence as smallest meaningful unit of text [6]. Sentences are the smallest coherent text unit and having meaningful message. As result converting sentences into numeric vectors is one of the most important proposed approaches. The study of automatic methods to represent words as numerical vectors has proposed by G.E.Hinton (1986) [7]. According to the first authors opinion of vector-based text processing, there are some important advantages of these methods that it is not possible by classical n-grams based methods [8]. Classic methods focus on contextual discrete units and they have no inherent relationship together. But in word vector-based models similar words in a one-concept class have very similar vectors.

The purpose of this investigation is to provide and explore a combined method which incorporates benefits of Google word2vec word vectors, text processing text processing and language models to convert sentences text to normalized and equal size vectors. This approach has results in covering all other approach shortcomings such as ignoring the feature of sentence length, different dimensional matrix size or generated sentences vectors with low difference. In the proposed method, the text processing preprocessing approaches are applied to input text and each sentences converted to unique lists. Then the sentences lists are developed to numerical matrices using word2vec algorithm and words mover distance (WMD) [9]. WMD applies words travel to convert sentences numerical matrices into similarity vectors. Finally, using most likely local n-grams, generated similarity vectors converted to equal and normal size. The unique size vectors are the best suitable data structure for most statistical linguistic technique and text processing algorithms.

The rest of this paper is organized as follows. Section 2 describes most important previous works on word embeddings and word2vec word converting method to vectors. Section 3 describes our proposed model including our best text preprocessing and converting sentences into normal vectors and finally conclusion is given in section 4.

2. Related works

The majority meaning of every text processing comes from word choice and the remaining comes from words order [10]. Word order is a crucial matter of capturing all the semantic information in sentence. Words can be mapped to vectors in a vector space. The mapping is called word embeddings and it is one of the most interesting areas in most of text processing approaches. Word embeddings specifies the meaning of each word in relation to other texts words. The idea of using other words in a text to understand the real meaning of each word was originally introduced by Ferret (1957) [11]. Fultz et al. introduced a vector-based model to calculate successive sentences semantic relation [12]. They used their vector-based model to evaluate text coherence. Connectionist approaches are one of the leading methods and proposing for developing distributed representations to encode the structural relationships between words [7] [13] [14]. There are some other proposed methods that employ word embedding to recognize document similarity [15]. Combining traditional method of corpus-based features and word embedding features are proposed by some other researchers like J. Tian and M. Lan [16]. They used some traditional NLP feature engineering of string-based similarity such as sentences length features, syntactic POS tagging features, longest common sequence, n-grams overlapping features and named entities features, machine translation similarity features, and combined with some new corpus-based features like *wordnet* rank features, vector space sentence similarity, global alignment and specific alignment features and word embedding feature engineering.

3. Word embeddings

Recent NLP approaches started with some methods based on just pure symbolic language analysis. Statistical methods on NLP task were introduced in 1980s and made NLP fields more popular. Most current NLP techniques assumed words as atomic units. Word level analysis lead to some advantages of simplicity, robustness and the observation of simple training models on huge amounts of data. However, traditional statistical simple techniques are at their limits in many tasks. For example:

Most of them were totally language dependent
Difficulty in detecting related and synonyms words
Difficulty in considering different writing forms of a word
No way to considering words location

In last decade other steps have been taken for mentioned shortcoming. They were proposing to represent and analyze words in vector spaces. Vector space representations have been successfully used in different syntactic and semantics NLP fields. In these approaches words could be mapped into vectors in a vector-space called word embeddings. Word embedding is tried to represent the usage, semantic and syntactic similarities, relatedness and position of words in sentence. On the other hand, word embeddings represent the meaning of words relative to other text words. The first work in which addressed word embeddings evaluation was the proposed method by Griffiths et al. (2007) [17]. In 2010 Turney and Pantel presented a comprehensive survey on word embedding [18]. Their proposed method used distributional semantics models which could be considered as a measure of word embeddings performance. A year after, D. S. McNamara was published the first comparison of performance on various DSMs [19]. He exemplifies that how statistical models of semantics play an important role in understanding of cognition and contribute to the field of cognitive science. In 2013, the popular word2vec tool was released by Mikolov et al. [8]. Word2vec is training on very large datasets and allows the model to learn complex word relationships such as:

$$vector(Berlin) - vector(Germany) + vector(France) \approx vector(Paris) \tag{3.1}$$

Vector space representations can be performed on two very close sentences or in a similar class [16]. Figure 1 presents the experimental words relation on two distinct sentences:

Storm will spread snow over Shanghai.
The earthquakes have shaken parts of Oklahoma.

In 2014, Baroni et al. proposed an extensive overview of approaches to word embeddings evaluation [20]. They presented the first systematic comparative evaluation of count and predict vectors. Faruqui et al. (2016) identified problems associated with word similarity evaluation of word vector models, and reviewed existing solutions wherever possible [21]. They suggest that the use of word similarity tasks for evaluation of word vectors can lead to incorrect inferences and calls for further research on evaluation methods.

4. Word2vec

Representing a text as numerically vectors and matrices, takes some advantage of vectors and matrix operators and lead to much more speed and accuracy. One of most famous and common

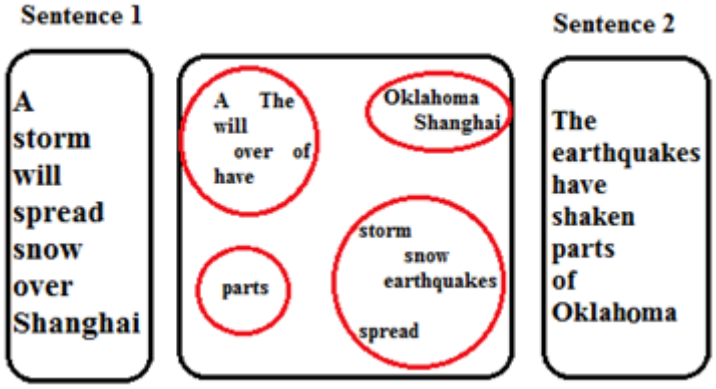

Figure 1: An Illustration of Word Embeddings [16]

vector-based text processing is Google word2vec algorithm [8]. The popular method generates elements vectors between -1 to 1 that are distributed numerical representations of the context of individual word features without human intervention (Equation 1). Where $nb(t)$ is the set of neighboring words and $p(wi|wt)$ is the maximum likelihood of travel cost of word wi to word wt . It makes highly accurate guesses about a words meaning based on its past appearances.

$$\frac{1}{T}\sum_{i=1}^{T}\sum_{j\in nb(t)}\log p(w_i|w_t) \tag{4.1}$$

Word2vec can be used in many text processing approaches such as sentiment analysis [22], text classification [23], automatic essay scoring [24], text simplification [25], text summarization [26], text generation [27], pattern matching [28] text coherence evaluation [29] natural language ambiguity [30] [31], email classification [32], searching similar texts [33], news offer [34] and similar goods. The algorithm uses neural networks and billions of words within several million web documents and generates a fixed size vector for each word in document. Each column in the vectors just shows a number and does not indicate a specific property. M. T. Abd1 and M. Mohd examined the prototypical word representation method and found that word2vec can be successfully used for Bio-Named Entity Recognition [35]. They show their word2vec vector representation method combined with two popular sequence-labeling approaches of Conditional Random Fields (CRFs) and Maximum Entropy Markov Models (MEMM) improved the performance of the two machine learning models with different datasets. We have an example of two sentences converting to unique matrix. If a 100 elements vector is generated for every word, we have a 100 dimensions space that every word in this space has a unique vector (table 1 and 2):

A storm will spread snow over Shanghai.
The earthquakes have shaken parts of Oklahoma.

Table 1 and 2 reveal the first seven elements of 100 elements vector of each sentences word. According to obtained word vectors, each sentence converts to unique matrix. But sentences with different length lead to different dimension of matrices. Some previous methods have attempted to make average value of columns and obtain a one-dimensional vector including N elements [36] [37].

Table 1: First Sentence Words Vector Generating by Word2vec

Words	**Vectors**							
a	-0.0145	-0.10315	0.09016	0.15967	0.12111	0.06169	-0.0506	
storm	0.27012	-0.11728	-0.10711	-0.17389	-0.11666	0.101391	0.297459	
will	0.01242	0.12589	0.00235	0.334008	0.052308	0.111021	0.118059	
spread	0.23484	-0.24044	-0.23968	-0.02178	0.209235	0.106129	0.003406	
snow	0.27845	0.08688	-0.02953	-0.21082	-0.15313	0.123246	-0.16418	
over	0.05527	-0.07588	0.22287	0.235038	0.390302	0.102481	-0.11601	
Shanghai	0.45604	-0.17005	0.13119	0.067335	-0.26712	-0.31781	0.33737	

Table 2: Second Sentence Words Vector Generating by Word2ve

Words	**Vectors**							
the	-0.16441	0.03211	0.08352	0.07601	0.14558	0.10881	0.04161	
earthquakes	0.07366	-0.01261	0.00131	0.02239	0.04506	-0.02498	0.01721	
have	0.05777	-0.09796	0.23698	0.05579	0.01641	0.00640	0.02438	
shake	0.03016	-0.13642	-0.31775	-0.06335	0.16948	-0.27697	0.05185	
part	0.10178	0.02292	0.07076	0.04385	0.17474	0.20870	0.14299	
of	-0.03745	-0.04726	0.28023	0.08130	0.01393	0.09037	0.08464	
Oklahoma	0.26341	-0.02099	0.00929	-0.04321	-0.05444	-0.21249	0.04522	

Averaging based generation sentences vector brings some advantages like simplicity of computing, generating an N element vector for all sentences, requiring low storage memory and more processing speed [38]. But the main challenge faced by them is:

Word location is an important feature in all text processing methods. In most of the times, changing the location of words leads to changing sentences information. But Changing location of words has no effect on output vector.

Ignoring sentence length feature. Sentence length is one of the most important and effective features in most text processing fields. In these approaches there is no difference between long and short sentences.

Because of positive and negative values in word2vec vectors, averaging based methods lead to create much closed vectors. So, sentence relationship and differences are having low accuracy. Table 3 presents the two-sentence averaging vector.

5. Proposed method

5.1. Text preprocessing

The first and most important issue in any text processing operation is preparing input text for applying the subsequent algorithms. The initial preprocessing is different according to text type,

Table 3: Sentences vector based on averaging methods

First sentence	0.184667	-0.07057	0.010041	0.055653	0.033722	0.041165	0.060776	
Second sentence	0.046422	-0.03717	0.052053	0.024686	0.07297	-0.01431	0.058275	

language, and main algorithm. Preprocessing types and the amount of its exploiting rate have huge impact on main algorithm speed and accuracy. Most important preprocessing algorithms are tokenizing, stop word removing, stemming, lemmatizing and POS tagging [39] [40]. However, their challenges are restricting to a particular filed, having no ability to apply and extend to all areas and languages. Two most important texts preprocessing are text cleaning and text normalizing. Samples for text cleaning are removing tags in html code, extra space and tabs and for text normalizing are normalized accent character (naive instead of nave) and informal writing text (hello instead of hellooo).

The main goal of our method is to convert all texts sentences into unique size numerical vector. Generated vectors are having the most sentence information and acceptable distance. Therefore, our method tried to equalize sentences vector length into two distinct phases of preprocessing and main algorithm. In our study, the first phase offers some important insights into preprocessing methodology and input text normalizing. At the beginning we apply text cleaning and word normalizing. Then we apply other common proposed preprocessing:

Sentence segmentation (tokenization): Firstly, using Python NLTK tools, sentences and words are separated. There is some limitation in default Python NLTK tokenizer. Therefore, in order to increase the precision of tokenizer, we taught a new tokenizer which is suitable for large texts. To learn new tokenizer we use PunktSentenceTokenizer tool and webtext corpus. WordPunctTokenizer tool is using for word tokenizing and sent-tokenize tool is using for sentence tokenizing.

Text cleaning: in text cleaning phase, first all uppercase characters converted to lowercase character. Then all extra space and characters are removing. These extra characters do not add any information to text ({}[]⌈⌉\/ ∩ ....)

Word normalizing: Accent characters are particular type of Unicode characters in borrowed words from different Latin form languages. In this phase we apply Python NLTK tools and wordnet database to convert words with accent characters and informal words (like hellooo) into Standard English characters. Word2vec database is a collection of 132400 words and their vectors, but there are possible words in the text that cannot be included in the database [41]. In this case, using wordnet database with NLTK tool in Python, the vector of word with largest common subclass is selected and placed in the sentence matrix. The other word normalizing which is applied to remaining words is lemmatizing. Lemmatizing and stemming are two important tools for converting words into basic and standard form. Stemming works based on word appearance (believes believ). But lemmatizing works based on word meaning and concept (believes belief).

5.2. Converting sentences into normal vectors

All sentences in documents have different length with different numerical matrix dimension. In order to apply text processing methods, sentences matrix or vector must be normalized. Some previous methods have attempted to make average value of columns and obtain a one-dimensional vector including N elements [36] [37]. Our previous study offers a simple and efficient method which covers some disadvantage of average base vectors [21] [38]. In our previous studies, instead of average word vectors, original matrices are using for subsequent text processing. But to optimize the shortcoming and making standard size matrices, instead of word vectors, using most likely n-grams, phrasal vectors are generated. In present study we propose a statistic and simple method that converted sentences text into numerical and normalized vectors. The aim of this study is to combines three most efficient methods (averaging based, most likely n-grams, and words mover distance) to use their advantages and reduce their constraints by overlapping of three fields. Our proposed method

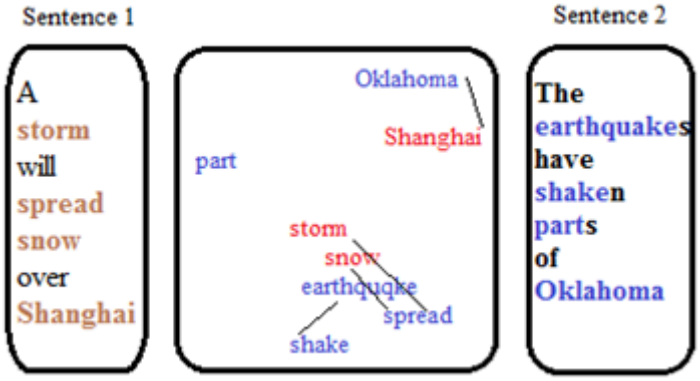

Figure 2: An Illustration of Word Movers Distance [16]

generates sequential sentence similarity vector instead of sentences matrix and local most likely n-grams to convert similarity vector to normal size. As result an N sentences document converted to (N-1) normalized vectors. To overcome the mentioned method and generate sentence similarity vector, our approach uses recent results of word2vec vectors [38] and Word Movers Distance (WMD), utilizes the property of word2vec embeddings [9]. WMD is incorporating the semantic similarity between individual word pairs into sentence distance metric.

One such measure of word similarity and dissimilarity is naturally provided by their Euclidean Distance (2), Inverse Cosine Similarity (3) and Hellinger Distance (4) in word2vec embedding space. The travel cost between two words is a natural building block to evaluate two sentences distance. In this method, first, the similarity between each word in first sentence is compared with each single word in followed sentence and then calculated their distance.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_1)^2} \tag{5.1}$$

$$d(p,q) = 1 - \frac{\sum_i \sqrt{p_i \times q_i}}{\sqrt{(\sum_i p_i^2)(\sum_i q_i^2)}} \tag{5.2}$$

$$d(p,q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}(\sqrt{p_i} - \sqrt{q_i})} \tag{5.3}$$

6. Experiments

For the first time text cleaning and text normalizing are applied on all sentences. Some sequential sentences and their preprocessing result lists are presented. Resulting lists are the outputs of applied preprocessing algorithms, including tokenizing, stop word removing and lemmatizing.

There was once a woman who wished very much to have a little child. She went to a fairy and said I should so very much like to have a little child.

A= ['there', 'woman', 'wish', 'much', 'little', 'child']
B= ['she', go, 'fairy', say, 'I', 'much', 'like', 'little', 'child']

Table 4: Words Mover Distance Matrix (WMDM), Calculating by Average of Three Criteria of Two Sentences

—	**she**	**go**	**fairy**	**say**	**I**	**much**	**like**	**little**	**child**
there	0.75657	0.64511	0.82923	0.85816	0.67067	0.65756	0.63365	0.46058	0.5883
woman	0.68376	0.67041	0.79336	0.89689	0.66131	0.59334	0.62293	0.47120	0.33549
wish	0.79769	0.68	0.90697	0.91313	0.72964	0.68617	0.65381	0.58297	0.58094
much	0.79275	0.73319	0.92328	0.94857	0.73339	0.08600	0.73815	0.43588	0.61045
little	0.64278	0.56185	0.75971	0.85559	0.59220	0.37685	0.69693	-0.03783	0.43344
child	0.73328	0.68328	0.80061	0.91689	0.70776	0.61674	0.67132	0.50661	0.02957
Average	0.73447	0.66230	0.83552	0.89820	0.68249	0.50277	0.66946	0.40323	0.42969

In our method, three mentioned measures are applied to make Words Mover Distance Matrix (WMDM) of words in lists A and B. As results sentence similarity matrix of two sequential sentences is generated (table 4). To convert sentence similarity matrix to sentence similarity vector, the average of each column makes an element of final vector:

[0.73447, 0.66230, 0.83552, 0.89820, 0.68249, 0.50277, 0.66946, 0.40323, 0.42969]

WMDM (1, 1) =dist (she, there)
WMDM (1, 2) =dist (go, there)
..
WMDM (6, 9) = dist (child, child)

To convert the sentence similarity vector into standard size, we use most likely local n-grams. To choose the best length for similarity vector we extract the minimum, average length of document sentences. The best length for similarity vector is the half-value of minimum and average sentences length. To normalize the sentence similarity vector length, we find the most probability local n-grams of two compared sentences and make phrasal n-grams. For example, in previous two sentences the most probability n-grams are $[much-like(0.586115), little-child(0.698385), she-go(0.41646)]$. As result the similarity vector converted to a six elements vector:

[0.698385, 0.83552, 0.89820, 0.68249, 0.586115, 0.41646]

Some time it is possible a document has two short sequential sentences. This situation leads to make similarity vector shorter than threshold. In order to solve the leading problem two sentences join to each other and assumed as one sentence. It can be each sentence join to its next or previous sentence.

7. Conclusion

The present study was proposed an efficient method for converting and normalizing sentenced into numerical vectors. Most text processing approaches that employ numerical sentence matrix suffer from unmatched matrices with different dimensions and tried to overcome its deficiency. We performed a careful empirical method which optimizes utilization of word vectors obtained word2Vec approach, statistical language models and text preprocessing likes stop word removing, stemming

and lemmatizing. The findings from this study make several contributions to the current sentence-based text processing methods. The modeling taken in this paper is relatively inexpensive, relies on statistical language models and basic text properties. This makes the proposed model particularly attractive for the automatic machine generated summaries, text coherence evaluation, and automatic essay scoring. We found that our consistent trend offers a simple and efficient architecture that lead to fast and high performance in compare to other post processing algorithms.

References

[1] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," arXiv preprint arXiv:1503.00075, 2015.

[2] H. He, K. Gimpel, and J. Lin, "Multi-perspective sentence similarity modeling with convolutional neural networks," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1576-1586.

[3] W. Ling et al., "Finding function in form: Compositional character models for open vocabulary word representation," arXiv preprint arXiv:1508.02096, 2015.

[4] M. Ballesteros, C. Dyer, and N. A. Smith, "Improved transition-based parsing by modeling characters instead of words with LSTMs," arXiv preprint arXiv:1508.00657, 2015.

[5] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Charagram: Embedding words and sentences via character n-grams," arXiv preprint arXiv:1607.02789, 2016.

[6] M. Abdolahi, M. Zahedi, A new model for text coherence evaluation using statistical characteristics Journal of Electrical and Computer Engineering Innovations, vol. 6, no. 1, 2018.

[7] G. E. Hinton, "Learning distributed representations of concepts," in Proceedings of the eighth annual conference of the cognitive science society, 1986, vol. 1, p. 12: Amherst, MA.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111-3119.

[9] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in International Conference on Machine Learning, 2015, pp. 957-966.

[10] T.K.Landauer A, psychology of learning and motivation, San Francisco: San Francisco, pp.43-84, 2002.

[11] R. Firth, "2. a note on descent groups in polynesia," Man, vol. 57, pp. 4-8, 1957.

[12] P. W. Foltz, W. Kintsch, and T. K. Landauer, "The measurement of textual coherence with latent semantic analysis," Discourse processes, vol. 25, no. 2-3, pp. 285-307, 1998.

[13] J. B. Pollack, "Recursive distributed representations," Artificial Intelligence, vol. 46, no. 1-2, pp. 77-105, 1990.

[14] J. L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," Machine learning, vol. 7, no. 2-3, pp. 195-225, 1991.

[15] S. P. Singh, A. Kumar, H. Darbari, A. Rastogi, S. Jain, and N. Joshi, "Building Machine Learning System with Deep Neural Network for Text Processing," in International Conference on Information and Communication Technology for Intelligent Systems, 2017, pp. 497-504: Springer.

[16] J. Tian and M. Lan, "ECNU at SemEval-2016 Task 1: Leveraging word embedding from macro and micro views to boost performance for semantic textual similarity," in Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), 2016, pp. 621-627.

[17] T. L. Griffiths, M. Steyvers, and J. B. Tenenbaum, "Topics in semantic representation," Psychological review, vol. 114, no. 2, p. 211, 2007.

[18] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," Journal of artificial intelligence research, vol. 37, pp. 141-188, 2010.

[19] D. S. McNamara, "Computational methods to extract meaning from text and advance theories of human cognition," Topics in Cognitive Science, vol. 3, no. 1, pp. 3-17, 2011.

[20] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors," in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2014, vol. 1, pp. 238-247.

[21] M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer, "Problems with evaluation of word embeddings using word similarity tasks," arXiv preprint arXiv:1605.02276, 2016.

[22] N. Rusnachenko and N. Loukachevitch, "Using convolutional neural networks for sentiment attitude extraction from analytical texts," EPiC Series in Language and Linguistics, vol. 4, pp. 1-10, 2019.

[23] R. A. Stein, P. A. Jaques, and J. F. Valiati, "An analysis of hierarchical text classification using word embeddings," Information Sciences, vol. 471, pp. 216-232, 2019.
[24] M. Cozma, A. M. Butnaru, and R. T. Ionescu, "Automated essay scoring with string kernels and word embeddings," arXiv preprint arXiv:1804.07954, 2018.
[25] H. Pylieva, A. Chernodub, N. Grabar, and T. Hamon, "Improving Automatic Categorization of Technical vs. Laymen Medical Words using Fast Text Word Embeddings," in 1st International Workshop on Informatics and Data-Driven Medicine (IDDM 2018), 2018.
[26] A. Jain, D. Bhatia, and M. K. Thakur, "Extractive text summarization using word vector embedding," in 2017 International Conference on Machine Learning and Data Science (MLDS), 2017, pp. 51-55: IEEE.
[27] Y. Zhu et al., "Texygen: A benchmarking platform for text generation models," in The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, 2018, pp. 1097-1100: ACM.
[28] S. Hakak, A. Kamsin, P. Shivakumara, and M. Y. I. Idris, "Partition based pattern matching approach for efficient retrieval of Arabic text," Malaysian Journal of Computer Science, vol. 31, no. 3, pp. 200-209, 2018.
[29] M. Abdolahi, M. Zahedi, "A new model for text coherence evaluation using statistical characteristics, " Journal of Electrical and Computer Engineering Innovations (JECEI), vol. 6, no. 1, pp. 15-24, 2018.
[30] R. A. Kadir, R. A. Yauri, and A. Azman, "Semantic ambiguous query formulation using statistical linguistical technique," Malaysian Journal of Computer Science, pp. 48-56, 2018.
[31] V. pal Singh and P. Kumar, "Nave bayes classifier for word sense disambigution of Punjab language," Malaysian Journal of Computer Science, vol. 31, no. 3, pp. 188-199, 2018.
[32] G. Mujtaba, L. Shuib, R. G. Raj, and R. Gunalan, "Detection of suspicios terrorist emalls using text classification: A review," Malaysian Journal of Computer Science, vol. 31, no. 4, pp. 271-299, 2018.
[33] J. Landthaler, B. Waltl, P. Holl and F. Matthes, "Extending Full Text Search for Legal Document Collections using Word Embeddings, " International Conference on Legal Knowledge and Information Systems, Sofia Antopolis, France, 2016.
[34] G. Jacobs, E. Lefever, and V. Hoste, "Economic event detection in company-specific news text," in The 56th Annual Meeting of the Association for Computational Linguistics, 2018, pp. 1-10: Association for Computational Linguistics.
[35] M. T. Abd and M. Mohd, "A comparative study of word representation methods with conditional random fields and and maximum entropy markov for Bio-named entity recognition," Malaysian Journal of Computer Science, pp. 15-30, 2018.
[36] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daum III, "Deep unordered composition rivals syntactic methods for text classification," 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), 2015, vol. 1, pp. 1681-1691.
[37] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Charagram: Embedding words and sentences via character n-grams," arXiv preprint arXiv:1607.02789, 2016.
[38] M. A. Kharazmi and M. Z. Kharazmi, "Text coherence new method using word2vec sentence vectors and most likely n-grams," in 2017 3rd Iranian Conference on Intelligent Systems and Signal Processing (ICSPIS), 2017, pp. 105-109: IEEE.
[39] S. Vijayarani, M. J. Ilamathi, and M. Nithya, "Preprocessing techniques for text mining-an overview," International Journal of Computer Science and Communication Networks, vol. 5, no. 1, pp. 7-16, 2015.
[40] M. J. Denny and A. Spirling, "Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it," Political Analysis, vol. 26, no. 2, pp. 168-189, 2018.
[41] https://developer.syn.co.in/tutorial/bot/oscova/pretrained-vectors.html.