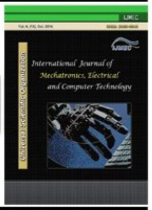
Vol. 8(30), Oct. 2018, PP. 4018-4028

Contents list available at IJMEC

International Journal of Mechatronics, Electrical and Computer Technology (IJMEC)

Journal Homepage: www.aeuso.org

PISSN: 2411-6173 - EISSN: 2305-0543



Sentence Matrix Normalization Using Most Likely N-grams Vector

Mohamad Abdolahi and Morteza Zahedi

Kharazmi International Campus Shahrood University Shahrood, Iran

Phone Number: *+9153140551,* ***Phone Number:*** *+9124738644*

****Corresponding Author's E-mail:*** *mabdolahi512@yahoo.com*

Abstract

Word embeddings is one of the most interesting natural language processing filed and has been shown to be a great asset for a large variety of NLP tasks. N-gram language models are also an important text processing methods. It is also based on statistics of how likely words are to follow each other. There is a growing body of literature that recognizes the combine two methods. However, big problem in all previous approaches based on word vectors are different size of sentences matrices. Some studies have shown the beneficial of calculating the average value of columns and creating a one dimensional vector including N elements.But the approach has some disadvantages such as its resistance against word ordering, ignoring sentence length feature and their vector sentences are very close to each other. This paper proposes a new methodology for introducing an efficient and very simple rich statistical model of word2Vec approach and n-grams language model to assess unique size sentence matrices. The unique size resulting matrix does not depend on the language and its semantic concepts. Our results demonstrate that certain models capture complementary aspects of coherence evaluation, text summarization, automatic essay scoring, detecting fake and copied texts, text topic comparison and thus can be combined to improve performance..

Keywords: *n-grams normalization,natural language processing (NLP), sentence matrix,text preprocessing,word2vec algorithm word vector.*

1. Introduction

The study of the automatic methods to represent words in numerical vectors has a long history [4]. According to the first author’s opinion of vector-based text processing, there are some important advantages of these methods that it is not possible by classical methods based on n-gram [18]. Classic models focus on contextual discrete units that have no inherent relationship together. But in word vector-based models similar words in a one-concept class have very similar vectors. For example, the distance between each singular to plural word, are equal for all singular and plural words:

X cars–X car ≈ X apples–X apple ≈ X books–X book

This distance is also approximately equal even for sentences and creates a vector of vectors (matrix):

X (The wall is blue) ≈ X (The ceiling is red)

Language models and n-grams are also a very effective tool in text processing. N-grams can be one word (unigrams), two words (bigrams), three-word (trigrams) or any other n-grams. N-grams are used as a sequence of words. These limitations are problematic in terms of theory and technology. There are some counterexamples of probability that is finding the next word needs a history of more words.

Article History:
Received Date: Jul. 11, 2018
Accepted Date: Sep. 15, 2018
Available Online: 12. Oct, 2018

However, this limitation imposes some other limitations to a statistical collection. Later, language models based on neural networks was proposed to improve the performance of classical n-grams language models [17], [20].

The purpose of this investigation is to provide and explore a combined method which incorporates benefits of both word vectors and n-grams approach. This approach has results in covering all other approach shortcomings. In the proposed method, first sentences are developed to numerical matrices using word2vec algorithm. Then the most likely n-grams are used to achieve unique dimensions matrix. Applying the mentioned method on input document, it is converted to the number of equal size matrices the same as number of text sentences. The matrices are ready for text processing algorithms such as extractive summarization, coherence evaluation, easy scoring and so on.

2. Related works

The majority meaning of a text in every language comes from word choice and the remaining comes from words order [16]. It seems quite important to considering word order as crucial matter of capturing all the semantic information in the sentence. One of the leading methods is connectionist approaches and it is proposed to develop distributed representations and encode the structural relationships between words [4], [5], [6]. A most important method proposed based on connectionist approach is a neural network language model [20]. The method uses a statistical language model to learn and representations altogether words to word vectors using all documents in web.

Some other approaches are inspired by this architecture and lead to proposing other developed methods. For example a neural language model is designed which eliminates the linear dependency on vocabulary size [11]. Another method is proposed a hierarchical linear neural model [1] and more recently, recurrent neural network architecture was suggested recurrent neural network and developed language modeling [19]. All mentioned architectures are being trained using statistical methods over large text corpora in web [16].

Word representation as continuous vectors is a very popular and effective method in all new Natural Language Processing approaches. Showing words with continuous vectors has a long history [4]. But recently a lot of works have been conducted to combine word representation and other language models. One of the popular models is the architecture of estimating language models by neural network, which is proposed by Y. Bengio [17]. The proposed method is used as a feed forward neural network and linear projection layer to learn the word vector representation and a statistical language model.

Word2vec is a new approach that recently has been attracted lots of attention. The method is introduced by T. Mikolov at 2013[18] and has been highly regarded in the past three years. It is two-layer neural network whose input is a text and the output is a collection of vectors derived from the properties of words in the text. In this way, each word displayed with a vector of numbers. Transforming words into vectors lead to performing mathematical operations on words and discovering their similarities. In this study, we also proposed a hybrid model of word2vec and n-grams language model. In the following parts we continue to introduce some proposed method based on word2vec.

Martin Andrews proposed a new GPU-friendly factorization method. The method tries to obtain a representation of annotation of being sparse and non-negative in each encoding dimension [8]. S. Rothe and H. Schutze presented a new word embedding calculus based on meaningful remainder subspaces. They applied the operations of the calculus for some text processing like antonym classification, polarity spectrum creation, morphological analogy and POS tagging. They also represent a new calculus for subspaces that supports operations like “(−1) × hate = love” [14]. M. J. Kusner and other his colleagues have presented a novel distance function between text documents. The method works based on recent results in word embeddings that learn meaningful representations for words from local co-occurrences in sentences semantically [9].

There are some proposed methods that employ word embedding to recognize document similarity [13]. J. Tian and M. Lan used the traditional NLP and corpus-based features for textual similarity estimation, as well as efficient word embedding features [7]. Other methods are proposed semantic textual similarity measures between pair of texts, even though the two texts with similar context is including and using different words. The methods attempted to incorporate the context space of the sentence from that sentence alone [3]. T. Kenter uses word embedding and describes a generic and flexible method for semantic matching of short texts [15].

3. N-grams

An n-gram model is a probabilistic language model for predicting the next item in such a sequence items. N-gram models are now widely used in probability, communication theory, computational linguistics, computational biology and data compression. Recently n-gram models are widely used in statistical natural language processing, speech recognition and statistical machine translation [8]. N-grams can also be used for sequences of words or almost any type of data such as large sets of satellite earth images and genetic sequence of DNA [2]. The simplicity and scalability are two benefits of n-gram models and algorithms using it. N-gram model is a kind of divide and conquer algorithm that divide a big problem into some small problems, solved the small problems, combine small solution and solve the main problem.

For the PC, provides authors with most of the formatting specifications needed for preparing electronic versions of their papers. All standard paper components have been specified for three reasons: (1) ease of use when formatting individual papers, (2) automatic compliance to electronic requirements that facilitate the concurrent or later production of electronic products, and (3) conformity of style throughout conference proceedings. Margins, column widths, line spacing, and type styles are built-in; examples of the type styles are provided throughout this document and are identified in italic type, within parentheses, following the example. Some components, such as multi-leveled equations, graphics, and tables are not prescribed, although the various table text styles are provided. The formatter will need to create these components, incorporating the applicable criteria that follow.

In text processing, sequence words are modeled in a way that each n-gram is composed of n words and predicts word xi based on xi-(n-1), xi-(n-2), xi-1. It is show that each word depends only on the last n-1 words. The assumption is crucial, because it is massively simplifies learning the language model problems from data. In addition, because of the growing and open nature of each language and its words, it is common to group new and unknown words in the language. An n-gram of size 1 is referred to "unigram", size 2 is a "bigram", size 3 is a "trigram" and larger sizes are sometimes referred to "four-gram", "five-gram", and so on.

N-gram models are often criticized for some drawbacks such as lack of any explicit representation of long range dependency. This is because the only explicit dependency range is (n−1) tokens for an n-gram model, and since natural languages incorporate many cases of unbounded dependencies. This means that only n-gram model by itself cannot determine distinguish unbounded dependencies. To cover the problem, we combined n-grams and another efficient text processing field word embeddings to concentrate sentence length by generating n-grans numerical vectors.

In 1996, R. Rosenfeld with the introduction of an idea in LD n-grams proved the information in the history of words declined with increasing distance between them and follows a specific pattern [12]. But with the gap more than five words; the available information will remain constant. He measures perplexity between distance words from one to ten. To prove his theory, he was determined the perplexity of words with thousand distances. In his investigations he found and proved there is no significant change in perplexity amount of six to thousand distances. The theory proved that all words in a text linked together and at distance of more than five, the relationship between words is approximately equal. Table 1 shows the results obtained from his investigate analysis of words distance perplexity and the best n-grams for assessing words relatenes are uni-grams to 5-grams.

Table 1: obtained perplexity of Long Distance Bigrams, traning on Brown dataset by million words

Distance	1	2	3	4	5	6	7	8	9	10	1000
Perplexity	83	119	14	135	139	138	138	139	139	139	141

4. Woed embeddings

NLP approaches started with methods based on just pure symbolic analysis of language. Statistical methods were introduced to NLP in its current form in the 1980s, and made NLP fields more popular. Most current NLP techniques treat words as atomic units. This choice has several good advantages such as simplicity, robustness and the observation that simple models trained on huge amounts of data. However, the simple techniques are at their limits in many tasks. For example:

- The model was totally language dependent
- Diagnosis and determine relevant words and synonyms are difficult
- Different forms of a word considered different
- The location of the word in sentence is less important
- Most importantly, a strong need to pre-processing and unifying words in writing and semantic

But in the last decade another step has been taken for the mentioned drawbacks and it was proposed to represent and analyze words in vector spaces Vector space representations have been successfully used in different syntactic and semantics NLP fields. In this approach words could be mapped to vectors in a vector-space called word embeddings [18]. The word embedding is intended to represent the usage, semantic and syntactic similarities, position in the sentence and relatedness of the words in the document they appear. On the other hand, word embeddings represents the meaning of the words relative to other words in the whole text.

By using the new word offset technique, simple algebraic operations can be performed on the word vectors, it was shown for example that vector results a vector that is very closest to the vector representation of the word Queen.

vector (King) – vector (Man) + vector (Woman) ≈ vector (Queen)

It does not just true for synonym words, the model can be performed for words in a similar class. For example matrices of two sentences are very close to each other:

("The wall is blue" and "the wall is red")

Further, it can be change multiple words:

("The wall is blue" and "the ceiling is red")

5. Word2vec

For most text processing techniques, it is necessary to display every word as a number or a numeric vector. Representing a text numerically takes advantage of numerical processing in machine learning algorithms. One of approaches that have recently become very common is word2Vec. This method was suggested by Google in 2013. The purpose and usefulness of word2vec is to group the vectors of similar words together in vector space. Word2vec generates elements vectors between -1 to 1 that are distributed numerical representations of word features, features such as the context of individual words without human intervention. It can make highly accurate guesses about a word's meaning based on past appearances.

The method is highly efficient and very popular. It can be used in many text processing approaches such as emotions assessing, searching similar texts, news offer, similar goods, mode detection, article automatic scoring, text simplification, text summarization, text generation and text coherence evaluation. The word2Vec algorithm uses neural networks and generates a fixed size vector for each word in the document. The vectors generated by applying billions words within several million web documents. Each column in the vectors just shows a number and does not indicate a specific property. For example, 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. For example we have two sentences and their vectors:

"The cat sat on the hat. The dog ate the cat and the hat."

Created vectors can be used to process sentences. Table 2 and 3 shows first four elements of 100 elements vector of each sentences word. According to the obtained vector of word a unique matrix is formed. But in this approach, the different size of a text sentences matrix is a big difficulty. As result, the imbalance in the first dimension leads to the complexity of the matrix comparison.The results obtained from the preliminary matice of two sentencessare presented in table 2 and 3. Some previous methods have attempted to make average value of columns and obtain a one dimensional vector including N elements. The results of their averaging methods om two sentences vector are presented in table 4 and 5.

Table 2: first sentence words vector generating by word2vec

	word	**vector**				
1	the	-0.16441	0.03211	0.08352	0.07601	…
2	cat	-0.27043	0.23799	-0.03023	0.264557	…
3	sat	0.32563	0.02780	-0.12020	0.362297	…
4	on	0.13098	-0.19419	0.04385	0.19398	…
5	the	-0.16441	0.03211	0.08352	0.07601	…
6	hat	0.37875	0.06620	-0.02849	0.06544	…
7	.	0.15080	-0.11711	-0.13287	0.18885	…

Table 3: second sentence words vector generating by word2vec

	word	**vector**				
1	the	-0.16441	0.03211	0.08352	0.07601	…
2	dog	-0.14440	0.11054	-0.01605	-0.20122	…
3	ate	0.055213	-0.01441	0.04162	-0.05403	…
4	the	-0.16441	0.03211	0.08352	0.07601	…
5	cat	-0.27043	0.23799	-0.03023	-0.26455	…
6	and	0.01490	-0.04335	0.05209	0.00551	…
7	the	-0.16441	0.03211	0.08352	0.07601	…
8	hat	0.37875	0.06620	-0.02849	0.06544	…
9	.	0.15080	-0.11711	-0.13287	0.18885	…

Table 4: first sentence words vector generating by averaging words vector

	word	vector				
1	the	-0.16441	0.03211	0.08352	0.07601	…
2	cat	-0.27043	0.23799	-0.03023	0.264557	…
3	sat	0.32563	0.02780	-0.12020	0.362297	…
4	on	0.13098	-0.19419	0.04385	0.19398	…
5	the	-0.16441	0.03211	0.08352	0.07601	…
6	hat	0.37875	0.06620	-0.02849	0.06544	…
7	.	0.15080	-0.11711	-0.13287	0.18885	…
Sentence vector		0.05527	0.01213	-0.02694	0.09972	…

Table 5: second sentence words vector generating by averaging words vector

	word	vector				
1	the	-0.16441	0.03211	0.08352	0.07601	…
2	dog	-0.14440	0.11054	-0.01605	-0.20122	…
3	ate	0.055213	-0.01441	0.04162	-0.05403	…
4	the	-0.16441	0.03211	0.08352	0.07601	…
5	cat	-0.27043	0.23799	-0.03023	-0.26455	…
6	and	0.01490	-0.04335	0.05209	0.00551	…
7	the	-0.16441	0.03211	0.08352	0.07601	…
8	hat	0.37875	0.06620	-0.02849	0.06544	…
9	.	0.15080	-0.11711	-0.13287	0.18885	…
Sentence vector		-0.03426	0.03735	0.01518	-003553	…

6. Advantages and disadvantages

This approach has the following advantages:

- Simplicity of calculation
- Creating a standard vector with N elements for all sentences within a document. Despite constant dimensions for all the sentences of document sentence, comparing, text processing and feature extracting, it is much easier than other methods.
- Lower memory requirements.

- More processing speed. The number of vectors generated in this way is equal to the number of sentences.

But this approach also has some disadvantages:

- Location of a word in the sentence caused little change in its meaning and as a result, the sentence concept. But Changing location of words has no effect on output vector.
- Ignoring the feature of sentence length. Sentence length is one of the most important and effective features in most text processing fields. This feature has not considered it mean vector approach and there is no difference between long and short sentences.
- Creating very close sentences vector to each other. There are positive and negative values in the vectors. The distribution of positive and negative vector elements is almost the same. So the obtained average sentences vectors are close to each other and finding the relationship between sentences is a little harder.

7. Pre-processing

The first subject matter to consider any text processing approach is preparing input text to apply text processing algorithms. Unfortunately, the words that appear in documents and queries often have many structural variants. So before applying any processing algorithm, some pre-processing techniques are needed to apply on any target document. The required pre-processing algorithm is highly dependent on the type of processing algorithm and input text. The suitable preprocessing algorithm converts the input text to standard form based on processing algorithm. The most important pre- processing methods are Tokenization, Stop word removal, Stemming and POS tagging for the text documents.

7.1. Preprocessing applied on our method

In this study some basic pre-processing will be done on the text and the input text will be ready for the extraction of n-grams and their vectors.

- Spacing between words and punctuation marks:

In standard text there is not any space between punctuation marks and its previous word. In n-grams, punctuation marks are considered as a separate member and in word2vec algorithm each of them has unique vector. So insert a space between word and punctuation marks is increasing the accuracy of any text processing.

- Removing extra spaces characters between words:

Deleting all extra spaces and converting all tabs to a standard space are the initial pre-processing in any text operation and they are also performed on our method.

- Do not convert uppercase to lowercase characters.

In most of text processing methods uppercase letters are converted to lowercase. But in our proposed method, this conversion will not be performed. A word may be started with a lowercase letter in mid-sentence and with uppercase in the beginning of the text. The differences show the location of word in sentence and so important. Also a word may be a noun in part text with uppercase starting. But the other part it is an attribute or adverb with lowercase starting. These words have different concept and meaning, converting them to same case, they will taking same concept and reducing accuracy of sentences concept. In Word2wec approach, word starting capital letter is different to same word starting with lowercase and their vector will be different. So to maximize accuracy, the conversion is not performing in our proposed method.

- Unification of accented characters.

It is one of the most important preprocessing and leads to unification of different form of writing and accents characters. The accented characters and the same outputs are:

Input: È, É, Ê, Ë, Û, Ù, Ï, Î, À, Â, Ô, è, é, ê, ë, û, ù, ï, î, à, â, ô

Output: e, e, e, e, u, u, i, i, a, a, o, e, e, e, e, u, u, i, i, a, a, o

- Select the nearest vector for words that do not exist in a database Vector.

There are some words in processing texts that they are not in the database vectors. To get the most similar word with statistical methods, the vector of word with largest common subsequence is selected.

8. Proposed method

In proposed method, instead of calculated average word vectors, original matrices are used for subsequent text processing. The proposed method has some advantages:

- Constructed matrices with acceptable distance.
- The probability of equality of two matrices is zero. Even by subtracting one word or displacement two words in sentence, matrix will be different.
- Averaging word vectors, most sentences' feature cannot be considered and extracted.

But the most important challenge of this method is that the matrices rows are not equal. Derived imbalance is due to different sentence size. To resolve the problems, some solutions can be proposed:

1. To equalize the size of sentences in the pre-processing phase. This normalization should not cause a change in the sentence concept. It is semantic normalization and has its complexity.
2. Create a matrix with the same words, but normalize the resulting matrix, containing almost all the features of the original matrix.

Following items can be offered for equalizing the size of sentences (first solution):

- Removing stop words in the sentence according to their importance, to make unique size sentences (Statistical approach).
- Dividing or combining sentences consecutively (Semantic approach).
- Ignoring the sentences boundaries, select a limit words as a single window for generating matrix. The limit can be selected based on some criteria such as average length of sentences, the smallest or highest sentence, and the total number of words of text (Statistical approach).

8.1. Using n-grams vector for sentences size normalization

The number of words that appeare in each document sentences are different. As a result, the number of rows in their numerical matrix are also different. To enhance the accuracy and simplicity of processing operations the number of rows should be fixed. Word2Vec approach only calculates word vectors and there is not any comment on phrase vectors or n-gram vectors. Very simply say, word2vec generates vectors of uni-grams. So to reduce the matrix rows for sentences with words more than threshold, n-grams vectors can be used. Bi-grams to 5-grams vectors can be obtained by combining and averaging uni-grams vectors. We employ more than one n-grams vector to normalize the larger sentences matrix. So, the matrixes can be All texts are not equal in sentences size. As a result, the number of rows in generated numerical matrix is different. Table 6 and 7 shows the proposed method to make semisize matices.

To enhance the accuracy and simplicity of processing operations the number of rows should be fixed. word2Vec approach only calculates word vectors and there is not any comment on phrase vectors or n-gram vectors. Very simply say, word2vec generates vectors of uni-grams. So to reduce the matrix rows for sentences with words more than threshold, n-grams vectors can be used. Bi-grams to 5-grams vectors can be obtained by combining and averaging uni-grams vectors. We employ more than one n-grams vector to normalize the larger sentences matrix. So, the matrixes can be obtained due to sentences length and defined threshold by combination of different n-grams vectors. For example by using more likely tri-grams vector, two sentences have equal size matrix.

The important point is even if we want to calculate the average vectors sentences the result vector will be equal to the previous method. In this case every sentence has a unique matrix but equal size to each other. The resulting matrix has changed even by changing the one word position and decrease the precision of internal sentence cohesion. The results obtained from the n-grams vector and make semisize sentences matrix are presented in table 6 and 7.

Table 6: first sentence matrix, generating by combining word2vec and n-grams model

	word	**vector**				
1	the	-0.16441	0.03211	0.08352	0.07601	...
2	cat	-0.27043	0.23799	-0.03023	0.264557	...
3	sat	0.32563	0.02780	-0.12020	0.362297	...
4	on	0.13098	-0.19419	0.04385	0.19398	...
5	the	-0.16441	0.03211	0.08352	0.07601	...
6	hat	0.37875	0.06620	-0.02849	0.06544	...
7	.	0.15080	-0.11711	-0.13287	0.18885	...

Table 7: second sentence matrix, generating by combining word2vec and n-grams model

	word	**vector**				
1	the	-0.16441	0.03211	0.08352	0.07601	...
2	dog	-0.14440	0.11054	-0.01605	-0.20122	...
3	ate	0.055213	-0.01441	0.04162	-0.05403	...
4	the	-0.16441	0.03211	0.08352	0.07601	...
5	cat	-0.27043	0.23799	-0.03023	-0.26455	...
6	and the hat	0.07641	0.01832	0.03570	0.04898	...
7	.	0.15080	-0.11711	-0.13287	0.18885	...

CONCLUSION

The present study was designed to determine a basic and efficient method to normalize numerical matrix sentences. most text processing approaches tends to overcome the deficiency of difference matrices size in sentences. Because of difference sentences length, the problem is appeared where the sentences are converting to numerical matrix. Our mentioned framework is optimize utilization of word vectors obtained word2Vec approach and n-grams. We argued that this representation is particularly suited for developing unique matrix size for every sentence in a document. The modeling taken in this paper is relatively inexpensive, relies on n-grams and basic text properties. The model found more likely n-grams and calculates their vectors by the average of the vectors of its constituent words. This makes the proposed model particularly attractive for the automatic machine generated summaries, text coherence evaluation and automatic essay scoring.

References

[1] A. Mnih, G. Hinton, "A Scalable Hierarchical Distributed Language Model", In Advances in Neural Information Processing Systems, vol. 21, pp. 1081-1088, 2008.

[2] A. Tomovic, P. Janicic, and V. Keselj, "n- Gram- based classification and unsupervised hierarchical clustering of genome sequences ", Computer Methods and Programs in Biomedicine, vol. 81, No. 2, pp. 137-153,2006.

[3] B. Ganesh, A. Kumar, "Semantic Relation from Word Embeddings in Higher Dimension", in proceedings of SemEval-2016, San Diego, California, Association for Computational Linguistics, pp. 706-711, 2016.

[4] G.E. Hinton, "Learning distributed representations of concepts", in proceeding of the English Cognitive Annual Conference of the cognitive Science Society, pp. 1-12, 1986.

[5] J.B. Pollack, "Recursive distributed representations", Artificial Intelligence-On connectionist symbol processing, vol. 46, pp. 77-105, 1990.

[6] J.L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure", Machine Learning, vol. 7, No. 2, pp. 195-225, 1991.

[7] J. Tian, M. Lan, “Leveraging Word Embedding from Macro and Micro Views to Boost Performance for Semantic Textual Similarity”, in proceedings of SemEval-2016, San Diego, California, Association for Computational Linguistics, June 16-17, pp. 621–627, 2016.

[8] M. Andrews, "Compressing Word Embedding", 23rd International Conference, ICONIP, Kyoto, Japan, pp. 413-422, 2016.

[9] M. Kusner, Y. Sun, “From Word Embeddings To Document Distances”, in proceedings of the 32nd International Conference on Machine Learning, Lille, France, . JMLR: W&CP, vol. 37, pp. 956-966, 2015.

[10] P. Koehn, “Statistical Machine Translation”, Published in the United States of America by Cambridge University Press, New York, 2011.

[11] R. Collobert, J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning", In International Conference on Machine Learning, ICML, pp. 160-167, 2008.

[12] R. Rosenfeld, “A Maximum Entropy Approach to Adaptive Statistical Language Modeling”, Computer Speech & Language, vol. 10, No. 3, pp. 187–228, 1996.

[13] S.P. Singh1, A. Kumar, and H. Darbari, “Building Machine Learning System with Deep Neural Network for Text Processing”, 2nd International Conference on Information and Communication Technology for Intelligent Systems, ICTIS 2017, Vol 84, pp.497-504, 2018.

[14] S. Rothe, H. Schutze, "Word Embedding Calculus in Meaningful Ultra dense Subspaces", ”, in proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, pp. 512-517, 2016.

[15] T. Kenter, M. de Rijke, “Short Text Similarity with Word Embeddings”, in proceeding of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, October 18 - 23, pp. 1411-1420, 2015.

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

[17] T. Mikolov, G. Zweig, “Context dependent recurrent neural network language model”, Spoken Language Technology Workshop (SLT), IEEE, pp. 234–239, 2012.

[18] T. Mikolov, K. Chen, G. Corrado, J. Dean, “Efficient estimation of word representations in vector space”, ICLR Workshop, CoRR, abs/1301.3781, 2013.

[19] T. Mikolov, M. Karafiat, L. Burget, J. Cer nocky, S. Khudanpur, "Recurrent neural network based language mode", ISCA, 26-30 September 2010, Makuhari, Chiba, Japan, pp. 1045-1048, 2010.

[20] Y. Bengio, R. Ducharme, P. Vincent, C. Janvin, "A neural probabilistic language model", Journal of Machine Learning Research, pp. 1137-1155,2003.

Autor(s)

Abdolahi was born in Mashhad, Iran, on October 22, 1964, Thursday. He is Ph.D. candidate in Shahrood University of Technology in the field of computer engineering - artificial intelligence. His is lecturer in Iranian Academic Center for Education, Culture and Research (ACECR), Mashhad, Iran. His special fields of interest are NLP, data mining, image processing and machine learning.

Morteza Zahedi graduated from the RWTH-Aachen University, Aachen, Germany and assistant Professor in Shahrood University of Technology. His special fields of interest are NLP, pattern recognition, image and video processing.