Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 10, No. 4, 2022, 505-514.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Benefiting from Structured Resources to Present a Computationally Efficient Word Embedding Method

Fatemeh Jafarinejad*

Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran.

Article Info

Article History:
Received 14 July 2022
Revised 21 September 2022
Accepted 25 September 2022

DOI:10.22044/jadm.2022.12113.2362

Keywords:
Word Embeddings, WordNet, Graph Embeddings, Node2Vec. Word Semantic Similarity.

**Corresponding author: jafarinejad@shahroodut.ac.ir (F. Jafarinejad).*

Abstract

In the recent years, new word embedding methods have improved the accuracy of NLP tasks. A review of the progress of these methods shows that the complexity of these methods is growing. Therefore, there is a requirement for methodological innovation to provide new word embeddings. Most current word embedding methods use a large corpus of unstructured data to train the word semantic vectors. The main idea of this paper is to directly use the knowledge embedded in the structure of structured data to introduce embedding vectors. Therefore, the need for high processing power, large amount of memory, and long processing time will be eliminated using structured resources, and conceptual knowledge hidden in them. For this purpose, a new embedding vector, Word2Node, is proposed. This method uses a well-known structured resource, the WordNet, as its training corpus. Our hypothesis is that it is possible to directly use the linguistic knowledge lies in WordNet's graphical structure to provide accurate and small embedding vectors. The evaluation of this method on the text classification task has shown that the proposed method works the same or better compared Word2Vec. This result has been achieved while the amount of training data has decreased by about 50000000%. Moreover, the comparison of the proposed method with some other knowledge graph based embedding methods indicates the superiority of the proposed method on the word semantic similarity task. These results show the capacity of structured data to improve the quality of existing word embedding methods and their resulting vectors.

1. Introduction

Utilization of machine learning methods in symbolic data structures (e.g. texts or graphs) requires the use of methods to convert these data structures into numerical data structures. Embeddings are one of the common methods that map the basic elements of their underlying system (for example, words or phrases in text, and nodes or edges in graph) to a point in an N-dimensional vector space. Capability of coding the semantic information of the basic elements and their conceptual relationship (for example, synonymy of words in a text or the adjacency of nodes in a graph) while mapping to a compact representation, has led to improved accuracy, speed, and memory consumption of machine learning models that use these vectors.

The word embedding methods have been shown to be effective for improving the performance of many tasks of Natural language processing (NLP). SENNA [1], Word2Vec [2], GloVe [3], BERT [4], RoBERTa [5], GPT-3 [6], and XLNet [7] are some examples of word embedding methods. Word embeddings have greatly improved the accuracy of NLP tasks, so that nowadays, the best systems in many tasks include these types of information. Text classification [8], sentiment analysis [9]–[11] machine translation (MT) [12], text summarization [13], image captioning [14],

and question answering [15] are some examples of these NLP tasks.

However, reviewing the progress of embedding methods shows that along with increasing the accuracy of NLP tasks, the new embedding methods have more and more training parameters. This forces the embedding methods to require larger training corpus, time and memory, and consequently, more powerful and expensive hardware for computation. Therefore, there is a need for methodological innovation to suggest strong and fast embedding methods.

Actually, the existing word embedding methods use a large corpus of unstructured texts and ignore the structure of input, if any. The novelty of this article is to introduce a new paradigm for creating word embeddings. The basic idea is to use the structure of structured texts to present the embedding vector, and thus reduce the size of the training corpus, the number of training parameters, training time, and memory.

We hypothesized that semantic information encoded within the structured resources could compensate for this data deficiency.

Albeit limited works have been done in literature in the field of embedding information of knowledge graphs (e.g. BabelNet [16], WordNet) as synset embeddings [17]–[20]. AutoExtend [17] finds the unknown WordNet synset embedding vectors from the known word embedding vectors of Word2Vec with huge computations using machine learning methods. Denis *et al*. [18] uses WordNet to proposed a synset embedding. They proceed by constructing some different similarity graphs over synsets using various synset similarity algorithms [21]. Their method learns word representation from synset embedding. Path2Vec [19] uses graph distance measures to propose node embeddings for WordNet nouns. Syn2Vec [20] builds a large-scale synset graph using different monolingual and cross-lingual colexification graphs, popular embeddings. They compute word embeddings from the embeddings of synsets using various fusion approaches.

In this paper, WordNet as a structured data is used for proposing a computationally efficient word embedding and test the performance of the basic hypothesis of the paper: graphical structure of the WordNet includes valuable linguistic knowledge that can be considered and not ignored to provide cost-effective and small-sized embedding vectors.

To take advantage of this graph, we went to a simple node embedding method, Node2Vec. Of course, different types of graph embedding methods can be used in this field. However, since the Node2Vec algorithm utilizes the Word2Vec method, by choosing this graph embedding method, a significant round trip has been done between the text and node embedding algorithms, which will show the high power of combinability of these methods together. Node2Vec generates node embeddings by applying Word2Vec to the corpus of graph random walks. Needless to say, newer types of word embeddings can also be used to construct node embedding using the corpus of random walks.

In this paper, WordNet is considered as the input graph. Words of WordNet are considered as nodes of WordNet graph. Applying the Node2Vec algorithm to this graph, suitable embeddings for words will result.

The evaluation of the proposed approach in two tasks of word similarity and text classification shows the efficiency of this method and its resulting embedding vector (named Word2Node) in comparison to its embedded basic embedding method (Word2Vec). Word2Node embedding vector is trained on PC hardware with little memory consumption in a short time. Furthermore, benefiting from a smaller number of elements in the embedding vector (70 instead of 300 elements in Word2Vec) will increase the speed and decrease the training parameters of the new embedding in the underlying machine learning tasks that will use it. In this work, we will analyze the efficiency of the proposed idea from these perspectives, as well.

The structure of the paper is as what follows. In Section 2, some theoretical backgrounds are explained. Graph embedding methods, word embeddings, and WordNet structure are briefly discussed in this section. The third section describes the proposed method. In Section 4, we will evaluate the proposed method in two tasks: word similarity and text classification. Lastly, we will conclude and describe the future work in Section 5.

2. Materials and Methods

2.1. Node embeddings

Graph embedding approaches, also known as network representation learning, embed some elements of the input graph (e.g. nodes [22], edges [23] or the entire or sub-graph [24]) into a continuous low-dimensional vector space. This mapping should preserve network properties so that, after an optimization step of the learned embeddings, geometric relationships in the embedding space reflect the structure of the original network [25]. Learned embeddings are then used as useful semantic-aware input to machine learning algorithms. Different graph

embedding approaches are categorized in [26] as: geometric embeddings [27], stochastic and probabilistic embeddings [22, 23], and neural network embeddings [24, 25].

Utilizing adjacency matrix of the graph, matrix factorization is one of the methods that can be used for graph embedding [26, 27]. Another way of thinking about graph embedding is to provide closer embedding vectors (e.g. higher cosine similarity) to similar element (e.g. nodes) of the underlying graph (according to some criterion). Adjacency-based similarity [22, 24, 28], multi-hop similarity [32], and closeness in random walks [23, 29] can be used in this regard to measure similarity among nodes. Random walk-based methods use some generated random walks (with different strategies, e.g. DFS walks, BFS walks or biased walks) to propose a node embedding.

Node2Vec [29] is a node embedding method in which Word2Vec [2] is applied to fixed-length random walks of graph. To do this, originating from each of the graph nodes, several biased random walks are produced. Thereafter, set of all walks are considered as sentences of a corpus whose words are the node names. Applying Word2Vec embedding on this corpus will result in the embedding of corpus words (nodes of the underlying graph). This method can be applied to directed/undirected graphs with/without weights. Walk length, number of walks of nodes, word2vec sliding window size, and the vector size are the hyper-parameters of this method.

2.2. Word embeddings

Compact representation of word embeddings allows machine learning methods to be used faster than spars matrices of one-hot or term frequency-inverse document frequency (tf-idf) vectorizations. Utilization of singular value decomposition (SVD) on matrix of point wise mutual information (PMI) or latent semantic analysis (LSA) [36] was one of the earliest methods to achieve this compact word representation. SENNA [1] is one of the first methods using neural networks to compute the word embeddings. The basic idea of it was to avoid task-specific man-made input features for NLP tasks (including part-of-speech tagging, chunking, named entity recognition, and semantic role labeling). Instead, it learns internal word representations on the basis of vast amounts of mostly unlabeled training data. SENNA requires about 200MB of RAM and should run on any IEEE floating point computer[1]. Word2Vec [2] uses a feed-forward neural network architecture with one hidden layer. Its 300-dimensional pretrained vector is trained on the Google News dataset (about 100 billion tokens, and 3 million word types). GloVe [3] learns to encode the information of the probability ratio of word co-occurrences in form of word vectors. It is trained on Wikipedia 2014 + Gigaword5 (with 6B tokens, 400K vocab). BERT [4] uses a bidirectional transformer to learn a language representation. 16GB of Books Corpus and English Wikipedia are used as training corpus in BERT and training time was 4 days using 4 TPU Pods.

Lately, several methods have been presented to improve BERT on either its prediction metrics or computational speed but not both. BERT uses masked language model (MLM), where only the masked tokens (15%) are predicted. To achieve better performance, XLNet [7] introduces permutation language modeling, where all tokens are predicted in random order. XLNet uses a larger data and more computational power to be able to do so. Actually, it was trained with over 130 GB of textual data and 512 TPU chips running for 2.5 days.

RoBERTa [5] introduces dynamic masking, so that the masked token changes during the training epochs. RoBERTa uses 160 GB of text for pre-training, which is 1000% more data than the data used in BERT. It uses 1024 V100 Tesla GPU's running for a day to get the pretrained embedding vector.

Therefore, it seems that so far most of the performance improvements of word embeddings are either due to increased data, computation power or training procedure. The need for faster inference speed tends to a new embedding: DistilBERT [37]. It uses the idea of distillation to approximate BERT's large neural network by a smaller one. However, it needs BERT's neural network to be trained firstly. It only uses half the number of BERT's parameters but retaining 95-97% performance of BERT in different applications.

In this paper, we address the idea of utilizing knowledge encoded in the structured data to faster develop an embedding vector in a most memory efficient manner.

2.3. Princeton WordNet

WordNets are one of the most important language resources created in different languages. They are

[1] https://ronan.collobert.com/senna/

structured data containing linguistic knowledge such as set of words, synonyms, antonyms, and taxonomic relations. In fact, the words, different meanings of them, and the set of synonymous words (synsets) can be considered as nodes of this lexical network. Relations between words or synsets form the edges of this graph, which encodes high-level information of the underlying language. Princeton WordNet [38] (simply called WordNet in this paper) was the first WordNet created for English. In WordNet, the relations between nodes include two categories: lexical relations (relations between words), and semantic relations (relations between two sets of synonymous words). Information related to words (in the 4 syntactic categories of noun, verb, adjective, and adverb) and their lexical relations (e.g. antonymy and derivation) are coded in this hyper-graph. Each word may have several senses depending on its different meanings. A word sense (or lemma) is a representation of one aspect of different meanings of a word. Set of synonyms form another important type of nodes in the hyper-graph of WordNet, called Synsets.

A synset contains all of the word senses with a specific meaning. Thus each synset has different lemmas, and is identified by its most commonly used lemma. Different types of semantic relations including hypernymy, hyponymy, meronymy, holonymy, and entailment form other types of edges of WordNet graph connecting two different synsets. Set of synsets and the most important semantic relation between them (i.e. taxonomic relations) form a lexical ontology of language.

Due to the high level linguistic information coded in WordNet lexical network, this lexicology have been used as one of the most important tools to solve many high level semantic problems of languages. Word sense disambiguation [39], sentiment Analysis [40], information retrieval [41], question answering [37, 38], word embedding vector construction [44], and machine translation [45] are some but not all applications of the WordNets. In this work, graphic structure of the Princeton WordNet is used to provide a small-sized cost-efficient embedding for words of language.

3. Theory and Calculations

3.1. Proposed idea

A review of the progress of word embedding methods shows that the complexity of these models, their training parameters, and therefore, memory consumption grows increasingly. Therefore, there is a need for methodological innovation for presenting new word embedding methodologies. We see that most current word embedding methods use a large corpus of unstructured data (and neglect corpus structural information, if any) to train the semantic vectors of words. The amount of corpus leads to more memory consumption and training time. If we could use a smaller corpus, the memory consumption and the training time will be reduced. The training corpus must contain some semantic information, and the proposed embedding method should use this semantic information to compensate the issue of small amount of training data. The basic hypothesis was that structure of structured data can provide us with this information. As a well-known structured data utilizing from semantic knowledge of human language, we went to WordNet. Therefore, the need for strong hardware, large amount of memory, and long processing time will be met using structures and conceptual knowledge lies in WordNet. In order to provide a novel word embedding method, we construct a graph from WordNet. By applying the Node2Vec algorithm to the constructed graph we produce embeddings for the graph nodes (i.e. all words of WordNet).

Therefore, we use Node2vec and Word2vec (that lies in Node2Vec) embedding methods to evaluate our idea of using semantic-aware training corpus to train word embeddings in a simpler manner (with less memory consumption and training time). We evaluate the resulting word embedding in two tasks and consider its effectiveness, while using simple embedding methodologies. Note that newer more powerful graph and word embedding methods can be used in this regard. Nevertheless, profit from small structured and semantic-aware training corpus to reduce amount of required processing power.

3.2. Methodology

The idea of using semantic-aware corpus to train word embeddings simpler is addressed in the previous section. Utilizing linguistic knowledge encoded in WordNet, we evaluated this idea. We need a method to benefit from this semantic information. Algorithm 1 outlined the procedure of resulting Word2Node embedding. Graphical structure of WordNet leads us to use a graph embedding technique (Node2Vec) in this regard. Princeton WordNet 3.0 [38] (simply referred to as WordNet) is used for this purpose. We firstly construct a graph from WordNet. The nodes of this graph correspond to the set of all words of the WordNet. The edges of the graph also show the direct/indirect relationships between the words. By direct relations we mean lexical relations that

already exist in WordNet and map two different words to each other. Indirect relations between two words, however, can be distinguished using a round trip to synsets containing a lemma of the words and semantic relations of them. Finally, applying the Node2Vec algorithm to the constructed graph will produce embedding for the graph nodes (i.e. all words of WordNet).

Algorithm 1: Word2Node: Node2Vec-based word embedding method using WordNet as semantic-aware training corpus

Inputs: Hyper-parameters $\mathcal{N}w$, $\mathcal{W}\ell$, d, w, df, sf. $\mathcal{N}w$: number of walks in Node2Vec, $\mathcal{W}\ell$: walk length of Node2Vec, d: dimension of embedding vectors, w: size of sliding window in Node2Vec and its corresponding Word2Vec algorithm, df, sf: Boolean flags set to True, if the graph is directed, and has self-edges, resp.

Output: Word embedding vectors

1. Compute $\mathcal{W}$, the set of all words of WordNet
2. Make a new directed/undirected (according to the value of df) graph $\mathcal{G}$ with $\mathcal{W}$ as set of its nodes
3. Compute the graph edges: Construct_GraphEdges($\mathcal{G}$,$\mathcal{W}$,df,sf)
4. Generating $\mathcal{N}w$ walks of length $\mathcal{W}\ell$ from the graph
5. Compute d-dimensional node embeddings with window of length $w\ell$ using Node2Vec algorithm

Algorithm 1 has two important phases: graph construction (lines 1 to 3), and calculation of embedding vectors using Node2Vec (lines 4 and 5). The procedure of computing the edges of the WordNet graph (line 3) is described in Algorithm 2. As mentioned before, edges of this graph could contain indirect relations of words indicating semantic relations of synsets containing words' lemmas, as well as direct lexical relations. For each word of WordNet as nodes, this procedure calculates these edges.

In lines 1.A.a-1.A.e, the relations between the synsets containing the lemma of each the senses of a word and lemma of their hypernym, entailment relation, hyponym, holonym, and meronym synsets are added to the set of graph edges. Note that since these relations are coded as semantic relations in WordNet, we must use different lemmas of these synsets (as a word) instead of the synsets themselves.

In line 1.A.f, calculation of the synonyms/antonyms of an adjective word is addressed. Note that a set of synonyms of a word can be calculated using lemmas of the synset containing the word. Moreover, antonymy relations are encoded as lexical relations in WordNet, and could be added to set of graph edges easily.

Suitable selection of WordNet graph structure as well as the hyper-parameters of Node2Vec (number of walks, walk length, embedding vector dimension, and sliding window size) affect the performance of the resulting embedding. We got them by trial and error, and investigated the effect of these hyper-parameter selections in the task of word similarity in Table 1 of the next section.

Algorithm 2: Construct _GraphEdges: Compute edges between WordNet words as nodes of WordNet graph, $\mathcal{G}$

Inputs: $\mathcal{G}$: the returned graph initially have nodes without any edges between them, $\mathcal{W}$: set of all nodes of the graph, df: a Boolean flag which is set to True, if the graph is directed, sf: a Boolean flag, set to True, if the graph has self-edges.

Output: WordNet Graph

1. For w in $\mathcal{W}$, set of all words of WordNet:
   - A. For s in set of all synsets of word w in WordNet:
     - a. for $\ell 1$ in the set of all lemmas of all hypernyms of s:
       - i. add the edge ($\ell 1$.name(), w) to the $\mathcal{G}$
       - ii. if df, add the edge (w, $\ell 1$.name()) to the $\mathcal{G}$
     - b. for $\ell 1$ in the set of all lemmas of all entailments of s:
       - i. add the edge ($\ell 1$.name(), w) to the $\mathcal{G}$
       - ii. if df, add the edge (w, $\ell 1$.name()) to the $\mathcal{G}$
     - c. for $\ell 1$ in the set of all lemmas of all hyponyms of s:
       - i. add the edge (w, $\ell 1$.name()) to the $\mathcal{G}$
       - ii. if df, add the edge ($\ell 1$.name(), w) to the $\mathcal{G}$
     - d. for $\ell 1$ in the set of all lemmas of all holonyms of s:
       - i. add the edge ($\ell 1$.name(), w) to the $\mathcal{G}$
       - ii. if df, add the edge (w, $\ell 1$.name()) to the $\mathcal{G}$
     - e. for $\ell 1$ in the set of all lemmas of all meronyms of s:
       - i. add the edge (w, $\ell 1$.name()) to the $\mathcal{G}$
       - ii. if df, add the edge ($\ell 1$.name(), w) to the $\mathcal{G}$
     - f. for $\ell\ell 1$ in the set of all lemmas of s:
       - i. add the edge (w, $\ell\ell 1$.name()) to the $\mathcal{G}$
       - ii. for $aa1$ in antonyms of $\ell\ell 1$:
         - Z. add the edge (w, $aa1$.name()) to the $\mathcal{G}$

4. Results and Discussion

In order to analyze the performance of the proposed idea and the new word embedding methodology, we evaluate it in the two tasks of word similarity and text classification. We compare the results of Word2Node embedding with the result of Word2Vec as its base word embedding method. For word2Vec, we download and use the pretrained 300-dimensional word vectors trained on Google news using Word2Vec CBOW algorithm [2]. Wordsim353 dataset [46] is used for word similarity task. Kaggle's News category dataset [47] and IMBD dataset [48] are used for text classification. The following provides results of applying Word2Node embedding on these datasets.

4.1. Word similarity task

In order to analyze the proposed method, we firstly evaluate the method and the effect of different hyper-parameters in word similarity task on Wordsim353 dataset [46]. This dataset consists of word pairs and their similarity measure (which is a number in range [0,10]). We divide similarity measures of this dataset by 10 to be in the range [0,1] and be comparable with positive cosine similarity.

To evaluate the embedding model, we get the embedding vector of each word pairs of the dataset, compute their cosine similarity, and compare it with the number suggested by the dataset itself (as mentioned, divided by 10). We do this procedure for Word2Vec embedding method, as well. Mean squared error (MSE) of the numbers suggested by each embedding method with the gold standard numbers is a measure of goodness of each embedding method. MSE of the proposed embedding method for an undirected graph with self-edges, and vector dimension 70, walk length 14, number of walks 50, and window size 14, was **0.0317**. MSE of Word2Vec on this dataset was **0.045**. These results are obtained while the proposed method use smaller vector size (70, instead of vector size 300 for Word2Vec). This experiment shows the effectiveness of the proposed word embedding method on task of word similarity. The effect of using other values for hyper-parameters in the proposed embedding is illustrated in Table 1.

Table 1. Evaluation of Structure-Aware WordNet Training Corpus and the Proposed Corresponding Embedding Method (Word2Node) in Word Similarity Task.

Method	*df*	*sf*	*d*	*Wl*	*Nw*	*w*	MSE
Word2Vec	-	-	300	--	--	--	0.123
Word2Node	F	T	70	14	50	14	0.063
(Proposed)	T	T	70	13	50	13	0.080
	F	T	100	17	60	10	0.081
	F	T	25	12	40	7	**0.049**
	F	T	40	12	40	7	0.053
	F	T	50	12	40	7	0.058
	F	T	50	5	40	7	0.060
	F	T	50	7	40	7	0.060
	F	T	70	14	50	10	0.066
	F	T	70	14	50	5	0.066
	F	T	70	14	50	7	0.063
	F	T	70	17	60	14	0.071

Furthermore, we evaluate the performance of the proposed word embedding method against three embedding methods [18]–[20], which used the information of knowledge graphs in word similarity task, as us. [18] reports the Pearson correlation of its method on various word similarity datasets (i.e. wordsim353 [46], RG [49], and MEN [50] dataset). We evaluate our method on another important word similarity dataset, MC [51], as well. Path2Vec [19] uses SimLex999 [52] dataset to evaluate the performance of its algorithm using Spearman correlation. It just uses 666 noun similarities of the dataset. For fairness of comparison, we just use noun concepts of the SimLex999 dataset, as well. Syn2vec [20] used the Spearman correlation as a criteria for performance evaluation on Multi-SimLex dataset [53]. As we use English WordNet, we compare methods just in monolingual English word similarities. Tables 2, 3 illustrate the results. Columns of these tables shows different datasets that are abbreviated as WS, RG, MEN, MC, SL99, and MuSL, respectively for wordsim353 [46], RG [49], MEN [50], MC [51], SimLex999 [52], and Multi-SimLex [53]. Moreover, Path2Vec, Word2Vec, and the proposed Word2Node methods are abbreviated in rows of tables as P2V, W2V, and W2N, resp.

Table 2. Comparing Pearson Correlation of some Methods on Word Similarity Datasets.

Method\ Dataset	WS	RG	MEN	MC	SL99	MuSL
[18]	0.52	0.76	0.31	-	-	-
W2V	**0.65**	0.77	**0.76**	0.79	0.46	0.44
W2N	0.48	**0.81**	0.54	**0.80**	**0.57**	**0.50**

Table 3. Comparing Spearman Correlation of some methods on Word Similarity Datasets.

Method\ Dataset	WS	RG	MEN	MC	SL999	MuSL
S2V [20]	-	-	-	-	-	0.47
P2V [19]	-	-	-	-	0.51	-
W2V	**0.70**	0.76	**0.77**	**0.80**	0.45	0.49
W2N	0.48	**0.79**	0.54	0.74	**0.56**	**0.50**

As it can be seen in these tables, the proposed method outperforms the two knowledge graph-based embedding methods in all of the datasets. However, for some dataset the Word2Vec embedding works better than the proposed Word2Node embedding.

4.2. Text classification task

We used part of the Kaggle's News category dataset [47] and IMBD dataset [48] to compare the effectiveness of the proposed embedding idea and method in the task of text classification. The Kaggle's News category dataset [47] contains 200853 news in 41 different categories. We

limited our work to three categories (name it as 3CATS-News): ENTERTAINMENT, POLITICS, and TECH. This subset of the database contains 50879 news items, 70% of which are used for training and 30% remained for testing. The IMDB dataset [48] is a dataset having 50K movie reviews for binary sentiment classification. A set of 40k movie reviews is provided for training. Validation and test both contain 5k reviews.

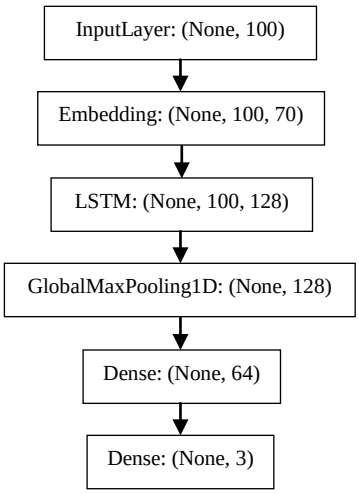

Figure 1. Architecture of the Used Classification Model.

The same deep learning architecture is used to evaluate the performance of Word2Vec and proposed embedding in text classification. This model is a sequential model containing an embedding layer, a LSTM layer, a max pooling layer, and two fully connected layers. Figure 1 shows the architecture of the model for Word2Node. In case of Word2Vec, the same model is used except the output dimension of embedding layer of which is equal to (None, 100, 300). In other words in the embedding layers, the application of pretrained Word2Vec weights is compared with using the proposed Word2Node embedding.

Table 2 demonstrates the accuracy and timing of each of the two mentioned embeddings. We consider 4 timing factors for each word embedding model: spent time to load the embedding model (load), time for construction of embedding matrix for deep learning models (const.), average training time of the deep model per epochs (train), and the prediction time (pred.) in seconds. As it can be seen in the table, the accuracy of the proposed method is approximately equal to that of the Word2Vec, while the predict speed is 23% faster than the Word2Vec method. Moreover, the loading time of the model and the construction of its embedding matrix are much less than Word2Vec. This difference in processing speed and time will be more pronounced in large datasets.

Table 4. Evaluation of Semantic-Aware WordNet Training Corpus and corresponding method, Word2Node on Text Classification Task.

Dataset	Method/Criteria	Accuracy	Load	Const.	Train	Pred.
3CATS-News	Word2Vec	93.933	35.7 98	0.16 93	51.5	7.21 73
	Word2Node on 3cats-news	93.436	2.43	0.03 12	34.1	5.50 16
IMDB	Word2Vec	87.60	35.7 98	0.25 78	200	2.48 13
	Word2Node	84.58	2.43	0.14 72	34	1.91 38

4.3. Cost analysis

In order to further evaluate the efficiency of the idea of utilizing a semantic-aware training corpus in embedding vectors, in this section, we will examine factors affecting the training cost of the proposed embedding method. The required processing power of the hardware used for training procedure is one of these factors. The amount of memory consumed and training time are some other factors that are of course related to the size of the training corpus. Table 5 shows the results of these factors. As it can be seen, utilizing structured data in presenting an embedding method will lead to a much reduction in the size required for the training corpus. In comparison to Google's pretrained vectors for Word2Vec, the required training data is reduced by about 50,000,000%. This factor will further affect the processing time, memory, and processing power.

Table 5. Required Processing Power to Train Word2Node Embedding Method.

	System Spec	Memory	Training Time	Training Corpus Size
Word2Node (proposed)	PC with Intel i7-8700K CPU @ 3.70GHz	8GB	14351seconds *(≈4hours)*	188097*

(*The number of words of WordNet used was 147,306. However, in terms of semantic relations and considering that we worked with synsets, some of lemmas of synsets do not exist in set of words of WordNet (inconsistency!) and added as nodes to the graph. Therefore, the number of final nodes of our graph is more than the number of words of WordNet).

5. Conclusions and Future Work

Nowadays, word embedding methods do a trade-off between prediction and computation metrics. Fundamental improvements that can increase performance while using fewer data and resources are required. In this paper, the basic idea of

utilizing from structure of the corpus was addressed. Utilizing linguistic knowledge encoded in WordNet, this idea was developed. Graphical structure of WordNet tends to produce word embedding vectors using a simple node embedding algorithm. Evaluation of the produced vectors in two tasks of word similarity and text classification shows the efficiency of this new paradigm shift in word embeddings. Faster training procedure and less memory consumption while are the results of this new embedding that it has an accuracy comparable to Word2Vec in these tasks. This idea can be more developed to extract word vectors using more powerful graph embedding and word embedding techniques. Nevertheless, benefit from small structured and semantic-aware training corpus to reduce amount of required processing power.

References

[1] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," J. Mach. Learn. Res., vol. 12, pp. 2493–2537, 2011.

[2] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and Their Compositionality," in Advances in Neural Information Processing Systems, Oct. 2013, pp. 1–9.

[3] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing ({EMNLP}), Oct. 2014, pp. 1532–1543, doi: 10.3115/v1/D14-1162.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." 2018.

[5] Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv Prepr. arXiv1907.11692, 2019.

[6] T. Brown et al., "Language Models are Few-Shot Learners," arXiv:2005.14165, May 2020.

[7] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," arXiv:1906.08237, Jun. 2019.

[8] R. A. Stein, P. A. Jaques, and J. F. Valiati, "An Analysis of Hierarchical Text Classification Using Word Embeddings," Inf. Sci. (Ny)., vol. 471, pp. 216–232, 2019, doi: https://doi.org/10.1016/j.ins.2018.09.001.

[9] Q. Chen and A. Crooks, "Analyzing the Vaccination debate in social media data Pre- and Post-COVID-19 pandemic," Int. J. Appl. Earth Obs. Geoinf., vol. 110, p. 102783, 2022, doi: https://doi.org/10.1016/j.jag.2022.102783.

[10] A. Pimpalkar and J. R. Raj R, "MBiLSTMGloVe: Embedding GloVe Knowledge Into the Corpus Using Multi-Layer BiLSTM Deep Learning Model for Social Media Sentiment Analysis," Expert Syst. Appl., vol. 203, p. 117581, 2022, doi: https://doi.org/10.1016/j.eswa.2022.117581.

[11] M. Molaei and D. Mohamadpur, "Distributed Online Pre-Processing Framework for Big Data Sentiment Analytics," J. AI Data Min., vol. 10, no. 2, pp. 197–205, 2022, doi: 10.22044/jadm.2022.11330.2293.

[12] E. Manzini, J. Garrido-Aguirre, J. Fonollosa, and A. Perera-Lluna, "Mapping Layperson Medical Terminology into the Human Phenotype Ontology using Neural Machine Translation Models," Expert Syst. Appl., vol. 204, p. 117446, 2022, doi: https://doi.org/10.1016/j.eswa.2022.117446.

[13] A. Joshi, E. Fidalgo, E. Alegre, and L. Fernández-Robles, "DeepSumm: Exploiting Topic Models and Sequence to Sequence Networks for Extractive Text Summarization," Expert Syst. Appl., vol. 211, p. 118442, 2023, doi: https://doi.org/10.1016/j.eswa.2022.118442.

[14] T. Xian, Z. Li, C. Zhang, and H. Ma, "Dual Global Enhanced Transformer for image captioning," Neural Networks, vol. 148, pp. 129–141, 2022, doi: https://doi.org/10.1016/j.neunet.2022.01.011.

[15] A. Shahini Shamsabadi, R. Ramezani, H. Khosravi Farsani, and M. Nematbakhsh, "Direct Relation Detection for Knowledge-Based Question Answering," Expert Syst. Appl., vol. 211, p. 118678, 2023, doi: https://doi.org/10.1016/j.eswa.2022.118678.

[16] R. Navigli and S. P. Ponzetto, "BabelNet: Building a Very Large Multilingual Semantic Network," in In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, 2010, pp. 216–225.

[17] S. Rothe and H. Schütze, "AutoExtend: Extending Word Embeddings to Embeddings for Synsets and Lexemes," in In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 2015, pp. 1793–1803, doi: 10.3115/v1/P15-1173.

[18] A. T. Thibault Cordier, "Learning Word Representations by Embedding the WordNet Graph," 2018.

[19] A. Kutuzov, M. Dorgham, O. Oliynyk, C. Biemann, and A. Panchenko, "Learning Graph Embeddings from WordNet-based Similarity Measures," in Conference: Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics, 2019, pp. 125–135, doi: 10.18653/v1/S19-1014.

[20] J. Harvill, R. Girju, and M. Hasegawa-Johnson,

"Syn2Vec: Synset Colexification Graphs for Lexical Semantic Similarity," in Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2022, pp. 5259–5270, doi: 10.18653/v1/2022.naacl-main.386.

[21] A. Budanitsky and G. Hirst, "Evaluating WordNet-Based Measures of Lexical Semantic Relatedness," Comput. Linguist., vol. 32, no. 1, pp. 13–47, 2006.

[22] Z. Zhao, X. Chen, D. Wang, Y. Xuan, and G. Xiong, "Robust Node Embedding Against Graph Structural Perturbations," Inf. Sci. (Ny)., vol. 566, pp. 165–177, 2021, doi: https://doi.org/10.1016/j.ins.2021.02.046.

[23] X. Wu, Y. Zheng, T. Ma, H. Ye, and L. He, "Document Image Layout Analysis Via Explicit Edge Embedding Network," Inf. Sci. (Ny)., vol. 577, pp. 436–448, 2021, doi: https://doi.org/10.1016/j.ins.2021.07.020.

[24] Q. Tian et al., "Lower Order Information Preserved Network Embedding Based on Non-Negative Matrix Decomposition," Inf. Sci. (Ny)., vol. 572, pp. 43–56, 2021, doi: https://doi.org/10.1016/j.ins.2021.04.095.

[25] A. Amara, M. A. Hadj Taieb, and M. Ben Aouicha, "Network Representation Learning Systematic Review: Ancestors and Current Development State," Mach. Learn. with Appl., vol. 6, p. 100130, 2021, doi: https://doi.org/10.1016/j.mlwa.2021.100130.

[26] L. Moyano, "Learning Network Representations," Eur. Phys. J. Spec. Top., vol. 226, pp. 499–518, 2017, doi: 10.1140/epjst/e2016-60266-2.

[27] G. Alanis-Lobato, P. Mier, and M. A. Andrade-Navarro, "Efficient Embedding of Complex Networks to Hyperbolic Space via Their Laplacian," Sci. Rep., vol. 6, no. 1, p. 30108, 2016, doi: 10.1038/srep30108.

[28] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-Scale Information Network Embedding," Line Large-Scale Inf. Netw. Embed., 2015, doi: 10.1145/2736277.2741093.

[29] A. Grover and J. Leskovec, "Node2vec: Scalable Feature Learning for Networks," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864, doi: 10.1145/2939672.2939754.

[30] D. Wang, P. Cui, and W. Zhu, "Structural Deep Network Embedding," in Proc. ACM SIGKDD, 2016, pp. 1225–1234, doi: 10.1145/2939672.2939753.

[31] Z. Zhang, P. Cui, and W. Zhu, "Deep Learning on Graphs: A Survey," IEEE Trans. Knowl. Data Eng., vol. PP, p. 1, 2020, doi: 10.1109/TKDE.2020.2981333.

[32] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning Graph Representations with Global Structural Information," in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, 2015, pp. 891–900, doi: 10.1145/2806416.2806512.

[33] J. Chen, Z. Gong, W. Wang, W. Liu, and X. Dong, "CRL: Collaborative Representation Learning by Coordinating Topic Modeling and Network Embeddings," IEEE Trans. neural networks Learn. Syst., vol. PP, Feb. 2021, doi: 10.1109/TNNLS.2021.3054422.

[34] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. Smola, "Distributed Large-scale Natural Graph Factorization," in WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web, 2013, pp. 37–48, doi: 10.1145/2488388.2488393.

[35] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online Learning of Social Representations," Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., Mar. 2014, doi: 10.1145/2623330.2623732.

[36] T. Landauer, P. Foltz, and D. Laham, "An Introduction to Latent Semantic Analysis," Discourse Process., vol. 25, pp. 259–284, 1998, doi: 10.1080/01638539809545028.

[37] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter," arXive:1910.01108, 2019.

[38] D. Soergel, WordNet. An Electronic Lexical Database. MIT Press, 1998.

[39] A. Pal and D. Saha, "Word Sense Disambiguation: A Survey," Int. J. Control Theory Comput. Model., vol. 5, Aug. 2015, doi: 10.5121/ijctcm.2015.5301.

[40] A. Montejo-Ráez, E. Martínez-Cámara, M. T. Martín-Valdivia, and L. A. Ureña-López, "Ranked WordNet graph for Sentiment Polarity Classification in Twitter," Comput. Speech Lang., vol. 28, no. 1, pp. 93–107, 2014, doi: https://doi.org/10.1016/j.csl.2013.04.001.

[41] O. El Midaoui, B. El Ghali, A. El Qadi, and M. D. Rahmani, "Geographical Query reformulation using a Geographical Taxonomy and WordNet," Procedia Comput. Sci., vol. 127, pp. 489–498, 2018, doi: https://doi.org/10.1016/j.procs.2018.01.147.

[42] T. Hao, W. Xie, Q. Wu, H. Weng, and Y. Qu, "Leveraging Question Target Word Features Through Semantic Relation Expansion for Answer Type Classification," Knowledge-Based Syst., vol. 133, pp. 43–52, 2017, doi: https://doi.org/10.1016/j.knosys.2017.06.030.

[43] S. K. Ray, S. Singh, and B. P. Joshi, "A Semantic Approach for Question Classification Using WordNet and Wikipedia," Pattern Recognit. Lett., vol. 31, no. 13, pp. 1935–1943, 2010, doi: https://doi.org/10.1016/j.patrec.2010.06.012.

[44] J. Goikoetxea, A. Soroa, and E. Agirre, "Bilingual Embeddings with Random Walks Over Multilingual

WordNets," Knowledge-Based Syst., vol. 150, pp. 218–230, 2018, doi: https://doi.org/10.1016/j.knosys.2018.03.017.

[45] D. Banik, A. Ekbal, P. Bhattacharyya, S. Bhattacharyya, and J. Platos, "Statistical-Based System Combination Approach to Gain Advantages Over Different Machine Translation Systems," Heliyon, vol. 5, no. 9, p. e02504, 2019, doi: https://doi.org/10.1016/j.heliyon.2019.e02504.

[46] L. Finkelstein et al., "Placing Search in Context: The Concept Revisited," ACM Trans. Inf. Syst., vol. 20, pp. 116–131, 2002.

[47] R. Misra, "News Category Dataset." 2018, doi: 10.13140/RG.2.2.20331.18729.

[48] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Jun. 2011, pp. 142–150, [Online]. Available: http://www.aclweb.org/anthology/P11-1015.

[49] H. Rubenstein and J. Goodenough, "Contextual Correlates of Synonymy," Commun. ACM, vol. 8, pp. 627–633, 1965, doi: 10.1145/365628.365657.

[50] G. Cassani and A. Lopopolo, "Multimodal Distributional Semantics Models and Conceptual Representations in Sensory Deprived Subjects." 2016, doi: 10.13140/RG.2.1.3394.3924.

[51] G. A. Miller and W. G. Charles, "Contextual Correlates of Semantic Similarity," Lang. Cogn. Process., vol. 6, no. 1, pp. 1–28, 1991, doi: 10.1080/01690969108406936.

[52] A. K. Felix Hill and Roi Reichart, "SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation," Comput. Linguist., vol. 41, no. 4, pp. 665–695, 2015.

[53] I. Vulić et al., "Multi-SimLex: A Large-Scale Evaluation of Multilingual and Crosslingual Lexical Semantic Similarity," Comput. Linguist., vol. 46, no. 4, pp. 847–897, 2020.

بهره مندی از منابع ساختاریافته برای ارائه یک روش کارآمد تعبیه کلمه

فاطمه جعفری نژاد*

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شاهرود، شاهرود، ایران.

ارسال ۲۰۲۲/۰۷/۱۴؛ بازنگری ۲۰۲۲/۰۹/۲۱؛ پذیرش ۲۰۲۲/۰۹/۲۵

چکیده:

در سالهای اخیر، روشهای جدید تعبیه کلمات دقت کاربردهای مختلف پردازش زبان طبیعی را بهبود بخشیده است. بررسی مقایسه‌ای این روشها نشان می‌دهد که پیچیدگی این مدلها و تعداد پارامترهای آموزشی آنها روبه‌رشد است. بنابراین، نیاز به نوآوری برای ارائه روشهای تعبیه کلمات جدید وجود دارد. اکثر روش‌های تعبیه کلمات فعلی از مجموعه بزرگی از داده‌های بدون ساختار برای آموزش بردارهای معنایی کلمات استفاده می‌کنند. ایده اصلی این مقاله، استفاده مستقیم از دانش تعبیه شده در ساختار داده‌های ساخت‌یافته برای معرفی بردارهای تعبیه کلمات است. بدین‌ترتیب نیاز به قدرت پردازش بالا، حجم زیاد حافظه، و زمان پردازش طولانی با استفاده از منابع ساختاریافته و دانش مفهومی نهفته در آنها برطرف می‌شود. برای این منظور، یک روش تعبیه کلمات جدید، Word2Node، پیشنهاد شده است. این روش از یک منبع ساختاریافته معروف، WordNet، به عنوان پیکره آموزشی خود استفاده می‌کند. فرضیه ما این است که می‌توان بدون واسطه از دانش زبانی موجود در ساختار گرافیکی WordNet برای ارائه بردارهای تعبیه‌شده دقیق و کوچک استفاده نمود. ارزیابی این ایده در طبقه‌بندی متون نشان داده است که روش پیشنهادی در مقایسه با روش تعبیه کلمات نهفته در آن (Word2Vec) یکسان یا بهتر عمل می‌کند. این نتیجه در حالی حاصل شده که حجم داده‌های آموزشی حدود ۵۰۰۰۰۰۰۰٪ کاهش یافته است. همچنین مقایسه روش پیشنهادی با برخی روشهای تعبیه کلمات مبتنی بر گراف دانش، در کاربرد تشخیص شباهت کلمات نیز حاکی از برتری این روش است. این نتایج نشان دهنده ظرفیت داده‌های ساخت‌یافته برای بهبود کیفیت روشهای تعبیه کلمات موجود و بردارهای حاصل از آنهاست.

کلمات کلیدی: تعبیه کلمات، وردنت، تعبیه گراف، Node2Vec، شباهت معنایی کلمات.