Using Graphs for Word Embedding with Enhanced Semantic Relations

Matan Zuckerman
Department of Software
and Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva 84105, Israel
matzuck89@gmail.com

Mark Last
Department of Software
and Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva 84105, Israel
mlast@bgu.ac.il

Abstract

Word embedding algorithms have become a common tool in the field of natural language processing. While some, like Word2Vec, are based on sequential text input, others are utilizing a graph representation of text. In this paper, we introduce a new algorithm, named WordGraph2Vec, or in short WG2V, which combines the two approaches to gain the benefits of both. The algorithm uses a directed word graph to provide additional information for sequential text input algorithms. Our experiments on benchmark datasets show that text classification algorithms are nearly as accurate with WG2V as with other word embedding models while preserving more stable accuracy rankings.

1 Introduction

Graph embedding in traditional studies aims to represent nodes as vectors in low-dimensional space. Nodes in a graph will have similar vectors if they share similar attributes. An example of a node attribute can be betweeness, which is defined as the number of times a node appears on the shortest path between two other nodes. Graph-based modeling has proved itself in many applications including node classification (Gibert et al., 2012), link prediction (Grover and Leskovec, 2016), community detection (Wang et al., 2017) and more (Goyal and Ferrara, 2018). While graph representation can be applied in a variety of domains, this paper will focus on natural language processing applications. Language can be represented by a graph of words in multiple ways. The node (word) connections can represent semantic relationships between words such as "king" and "queen", different grammatical forms of the same word such as *walk* and *walked*, co-occurring words, etc. Semantic similarity, which is an inherit part of word embedding, can help with a variety of NLP applications, such as text summarization (Nallapati et al., 2016), document classification (Yang et al., 2016), and machine translation (Zou et al., 2013). Creating a good feature representation of words is crucial for achieving reasonable results in all these tasks.

In this study, we propose a novel word embedding algorithm named WordGraph2Vec, or WG2V. WordGraph2Vec combines the benefits of existing word embedding algorithms and tries to minimize their downfalls. While existing algorithms are focused on input in the form of vectors alone or in the form of graphs alone, WordGraph2Vec takes into consideration both of these input types in its training phase. The word graph construction procedure used by WordGraph2Vec is similar to the graph representation in Schenker's (2003) work, where each unique word is represented by a node in the graph and there is a directed edge between two nodes if the corresponding two words follow each other in at least one sentence. The difference is that Schenker's original word graph was used for representing individual web documents whereas WordGraph2Vec generates a word graph from a large corpus of text that is expected to represent a natural language. We evaluate WordGraph2Vec vs. the state-of-the art node embedding algorithms (Grover and Leskovec, 2016; Tang et al., 2015; Perozzi et al., 2014) on the tasks of semantic relationships detection and text classification.

2 Related Work

In this section, we cover the related work on text representation and the benefits of graph-based representation models for natural language processing, along with common word embedding methods and state-of-the-art algorithms for graph embedding.

Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13), pages 32–41
Hong Kong, November 4, 2019.

2.1 Text Representation

One of the challenges in natural language processing is how to represent a term. The most common term representation is vector representation, such as "one-hot" vector, a vector in the size of the vocabulary, where only one dimension is equal to one and all others are zeros. More advanced methods of embedding terms (mostly, words) in a vector space are covered in the next sub-section.

Graph representation of text, by its natural structure, defines relationships between graph nodes. Each graph node represents a term, which can be defined in various ways including words, sentences, and n-grams. The node connections can define the "closeness" of terms to each other in a richer way than the "one-hot" vector representation including lexical and semantic relations, contextual overlap, etc. A potential advantage of using a directed graph model over context-dependent representations, such as word2vec, is preserving information about the word order in the input text.

Graph representation is not new in the world of text processing (Schenker et al., 2005; Sonawane and Kulkarni, 2014). Graph representations outperformed the classical vector representations of text documents, such as TF-IDF, on several NLP tasks including document classification (Markov et al., 2008), text summarization (García-Hernández et al., 2009), word sense disambiguation (Agirre and Soroa, 2009), and keyword extraction (Litvak and Last, 2008).

2.2 Word Embedding

Word Embedding is the process of representing words as dense vectors in a low-dimensional space. This process gained momentum since the paper of Mikolov (2013), which proposed a novel deep learning algorithm that yields word embedding. Mikolov's Word2Vec algorithm can be implemented in two ways. The first one is CBOW, predicting a word based on the words surrounding it. The second one is Skip-Gram, predicting the surrounding words of a specific word based on this word. The target word and its neighbors are selected from a sliding window, which traverses the corpus in linear steps. Similar words that do not fall in the same window, because they do not appear next to each other in the corpus, might not be "close" to each other in the low-dimensional space as will be shown in the next section. The Word2Vec training objective, as implemented in Skip-Gram, is to maximize the average log probability of:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \qquad (1)$$

Where T is the entire input text, c is half of the window size, and w are all co-occurring words. GloVe (Pennington et al., 2014) is another algorithm, which produces word embedding. The algorithm is essentially a log-bilinear model. The model tries to minimize the discrepancy between the estimated probability of two words appearing next to each other and the true probability of their co-occurrence. Both GloVe and Word2Vec are considered state-of-the-art algorithms for creating word embedding. Those word embedding methods are suffering from the limitation of ignoring the high-level, conceptual context of a given word. The vector of each word is trained only on its raw, low-level context and not on relations of words out of that context, which may affect the quality of word embedding. The word order is also ignored by the above methods.

2.3 Graph Embedding

Graph embedding models can overcome the limitations of the sequential input methods mentioned in the previous section. Paths in a word graph may connect semantically related words, which do not necessarily appear in the same low-level context. Those connections can be utilized for enriching the word embedding. Thus, a graph-based model can take into account both the structural and the semantic information represented by text.

An example of the above can be demonstrated by the following two sentences from a given corpus: "The boy went to school" and "The man went to work". If the sentence "The boy went to work" does not exist in the corpus, a sequential input model, such as Word2Vec, would not be trained on it, whereas a graph input algorithm can also be trained on this word sequence, because the node "went" is shared in the graph between the two sentences. Since "The boy went to work" is a valid path in the graph, the corresponding sentence can be included in the training set. This way we can enhance the semantic relationships between words represented by the embedded vectors. In addition, there is a need for NLP algorithms that can exploit the benefits of the graph input on one hand and face the challenges it brings on the other hand.

There are several state-of-the-art algorithms for producing graph embedding. Graph embedding can be node embedding, edge embedding or both. When defining an embedding, the "similarity" between nodes is the main key. In a graph representation, there are two major node similarity measures that can be applied:

- Nodes with the same neighbors are logically supposed to be similar. For example, in a social network a friend of my friend has a high probability to be my friend.
- Nodes that are not connected to each other, but have the same structural attributes, such as hubs.

Different algorithms approached those similarities in different ways. DeepWalk (Perozzi et al., 2014) is an algorithm based on the notion of a random walk. Random walk can be used to approximate similarity (Fouss et al., 2007). It is a method which is useful in large graphs when it is not possible for a computer to handle all of the graph data in its memory. DeepWalk algorithm tries to maximize the log probability of observing the last k nodes and the next k nodes in a random walk. The length of the random walk is set as 2k + 1. The algorithm generates multiple random walks and it tries to optimize the sum of log-likelihoods for each random walk. Random walks, as opposed to the Word2Vec model, do not scan the data linearly, which can be beneficial if the data is not linear. (Perozzi et al., 2014) did not evaluate the DeepWalk algorithm on text data.

Graph embedding algorithms achieved superior performance in domains that can be naturally modeled as graphs including social networks, article citations, word graphs, etc. A sample application is link prediction in social networks (Grover and Leskovec, 2016), where the Node2Vec algorithm achieved a high AUC score.

The Node2Vec algorithm is also based on a random walk. The algorithm tries to maximize the occurrence of subsequent nodes in a random walk. The main difference between the Node2Vec and DeepWalk is that Node2Vec has a mechanism of trade-off between breadth first search (BFS) and depth first search (DFS). Each search will lead to a different embedding. While in BFS the similarity is mostly based on neighboring nodes, in DFS further nodes will have a higher impact and hence will succeed to represent the community better. This trade-off produces better, more informative and higher quality embedding. An example of using Node2Vec algorithm for text representation can be found in Figure 1.

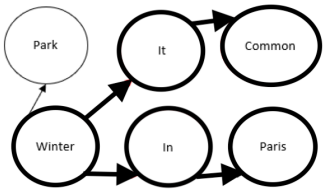

Figure 1: Example for Node2Vec walks. The nodes in bold represent a valid walk that was generated from Node2Vec algorithm. A sliding window will be trained on this walk.

The text graph used by Node2Vec in (Grover and Leskovec, 2016) is different from Schenker's model. The Node2Vec graph is undirected. Each two co-occurring words in the text have an edge between them. The paper does not mention whether they applied a minimal word frequency threshold to graph nodes or took into account sentence boundaries when defining edges.

Another node embedding algorithm is LINE (Tang et al., 2015). LINE algorithm is based on two types of connections:

- first-order proximity - local pairwise proximity between two nodes
- second-order proximity - proximity between the neighborhood structures of the nodes

The algorithm objective is to minimize the combination of the first and the second-order pairwise proximity. LINE defines for each two nodes their joint probability distributions using the two proximity measures and then minimizes the Kullback-Leibler divergence of these two distributions. LINE was evaluated on a text graph. In LINE, the text graph is undirected, words with frequency smaller than five are discarded, and words are considered as co-occurring if they appear in the same sliding-window of five words. It was not mentioned in the paper whether the graph edges were affected by the sentence boundaries.

Both DeepWalk and Node2Vec have the limitation of skipping some nodes by the random paths the algorithms generate for training. Consequently, some semantically related words may never appear on the same path. The LINE algorithm only considers first and second order relations between words and ignores the relations of

a higher order. This restriction may create word embeddings with lower quality.

3 Methodology

To overcome some of the limitations of existing word embedding methods, we propose a new, graph-based methodology, which has the following original contributions:

- Word graph construction: utilizing Schenker's directed graph model and extending it to a large language corpus.
- Graph-based word embedding: introducing a new algorithm named WordGraph2Vec, which combines the advantages of the Word2Vec and Node2Vec algorithms.

3.1 Word Graph Construction

The construction of the text graph in this paper is similar to Schenker's work (Schenker et al., 2005), but Schenker's graphs were constructed from relatively short web documents rather than from a large corpus representing an entire language. Each unique word in the corpus becomes a node in the graph and there is a directed edge between two nodes if the corresponding two words are consecutive in at least one sentence in the corpus. Stop words and words with less than 100 occurrences across the corpus were removed. The graph contains only words and not punctuation marks of any kind. The edge label represents the number of times the second word follows the first one. The graph is directed and weighted by the co-occurrence frequency. We preserve the edge directions, because the order of words in a sentence is usually important for its meaning, which should also affect the calculation of edge frequency weights. For example, the phrase "hot dog" will be represented in the graph with a relatively higher weight than "dog hot" which might have an edge in the graph but with a much lower weight as this phrase is very rare in English language. The text graph we use is different from the graphs in LINE and Node2Vec algorithms, which were discussed in the Related Work section.

3.2 WordGraph2Vec Algorithm

WordGraph2Vec builds upon the Word2Vec (Mikolov et al., 2013). The main difference between WordGraph2Vec to Word2Vec is that WordGraph2Vec enriches the text by adding target words for a specific context word in a sliding-window. In each sentence, m random context words are chosen. From the sliding window, which contains the context words, t target words are chosen. For each target word, n neighboring words in radius r are picked from the graph and added as a new context-target combination. Here the training objective is to maximize the log probability of words in the same sliding window and the log probability of the new context-target combination. This is done by maximizing equations 1 and 2.

$$\frac{1}{T'}\sum_{t=1}^{T'}\sum_{t\in m, nt\in n}\log p(w_{nt}|w_t) \qquad (2)$$

where t indices refer only to the words that were chosen as the m context words and nt are the new target words from the graph that are relevant to a specific t. T' is the number of new word combinations. In this way, the context words can predict additional target words from the word graph that do not necessarily appear in the same sentence but share a semantic relation to the context words.

Algorithm 1 WordGraph2Vec algorithm

Input: M: Number of context words, T: Number of target words, N: Number of neighboring words, R: Radius in graph, Text: Text, Sliding-Window: Size of the sliding window, G: Words Graph, E: embedding size

Output: Word Embedding for each node $v \in G$

for $Sentence \in Text$ **do**
  $CWords \leftarrow SelectCWords(Sentence, M)$
  for $SlidingWindow \in Sentence$ **do** Word2Vec ($SlidingWindow$, E)
    for $word \in SlidingWindow$ **do**
      if $word \in CWords$ **then**
        $TargetWords \leftarrow SelectTWords(SlidingWindow, T)$
        for $target \in TargetWords$ **do**
          $graphWords \leftarrow SelectNWords(N, target, G, R)$
          for $node \in graphWords$ **do** Word2Vec (combination of $node + word$, E)

Figure 2 shows an example of WordGraph2Vec algorithm where the radius is one. Two context

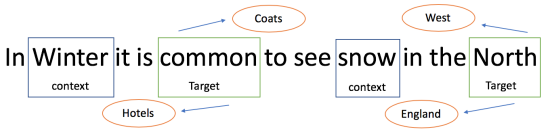

Figure 2: Example for WordGraph2Vec algorithm.

words are chosen and for each context word, one target word is picked from a specific sliding window in the size of five. For each target word, two new target words are picked from the graph. The advantage of WordGraph2Vec over Word2Vec and Node2vec is that it overcomes the limitations of both models. While the Word2Vec model only looks at linear relations between words, our algorithm takes into account context-target word pairs that are not necessarily taken from existing sentences in the corpus. In contrast, Node2Vec samples a path of words from the word graph using random walks rather than considering all neighboring words of a specific word. This means that Node2Vec, unlike Word2Vec, may ignore semantic relations between some co-occurring words in the original text.

The Algorithm 1 pseudocode presents the proposed algorithm (WordGraph2Vec). The *SelectCWords* function selects C random context words from a sentence. *SelectTWords* selects T random target words from the sliding window where at least one word is in *CWords*. *SelectNWords* selects N additional target words from the graph within the radius R. Word2Vec is trained combinations of context and target words for embedding of size E.

4 Empirical Evaluation

In this section, we discuss the evaluation metrics, the data, and the algorithms used in our experiments.

4.1 Language Corpus

The Wikipedia dump [can be downloaded from `https://dumps.wikimedia.org/backup-index.html`] dataset is a publicly available resource. Wikipedia publishes every couple of months a free xml file for all their articles and metadata in several languages. The Wikipedia dump used in this paper is in the English language. The dump was created on "2018-07-20" and consists of approximately 5,700,000 articles. The dataset contains more words than exist in the English language as there are also words that represent proper names such as locations, people, organizations, dates etc. The Wikipedia corpus is a collection of millions of sentences, which are expected to represent well the real-world distribution of a language.

Parameter	Value
Tokens	2,183,079,274
Unique words	13,744,507
Tok' less than 5 incidence	11,430,003
Tok' less than 100 incidence	13,436,432

Table 1: Wikipedia dump statistic.

While there are many unique words in the Wikipedia dump, only a small amount of them is frequently used. Thus, 83.1% of the words appear in the corpus less than five times as can be seen in Table 1 and only 2.2% of the words appear more than 100 times. This small group of 2.2% unique words is responsible for 97.2% of all word occurrences in the entire corpus.

In the view of these findings, we implemented the graph construction method introduced by Schenker (Schenker et al., 2005). All words with frequency of lower than 100 times were discarded. In addition, stopwords, as they usually do not contribute to the semantic meaning of other words, were also removed. Punctuation marks were removed from the graph as well.

4.2 Evaluation Tasks and Metrics

Evaluation of the word embedding methods was performed on two main tasks:

- Analogy test, both by Gladkova (analogy test, a) and Mikolov (analogy test, b)
- Document classification, based on benchmark datasets

4.2.1 Analogy test

Both Gladkova (2016) and Mikolov proposed analogy tests, which aim to test the semantic relations between "similar" words. The main difference between the two tests is in the combination of the questions. The test Gladkova proposed contains 99,200 analogy questions in 40 morphological and semantic categories. The 40 linguistic relations are separated into four types of relations: inflectional and derivational morphology, as well as lexicographic and encyclopedic semantics. Examples of these types can be seen in Table 2.

Types	Examples
Inflections	car:cars, go:going
Derivation	able:unable, life:lifeless
Lexicography	sea:water, clean:dirty
Encyclopedia	spain:madrid, dog:bark

Table 2: Examples for the four different types of analogy questions

All four types are balanced, i.e., there is the same number of questions for each type. Mikolov proposed a test which contains 19,544 questions from 14 different relations. The amount of questions is unbalanced across the relations. For example, the country:capital relation appears in over 50% of the questions. All questions are in the structure of a is to b as c is to d. Solving analogy questions can be done using the similarity between words calculated by equations 3 and 4.

$$d = argmax_{d \in V} \ (sim(d, c - a + b)) \qquad (3)$$

where V is a set of all words in the vocabulary excluding a,b and c, and sim is defined as:

$$sim(u, v) = cosine(u, v) = \frac{(u * v)}{||u||||v||} \qquad (4)$$

An answer to a question was labeled as correct if the correct word was among the top ten most similar words in the returned results.

Each question was tested on the word embedding vectors that were generated by the different algorithms, which are described in subsection 5.1. For each algorithm, the percentage of questions that were answered correctly was presented as the accuracy of each word embedding method.

4.2.2 Document classification

Document classification is a common NLP task. While the application of this task is well understood, the implementation of classification algorithms can be difficult. Using word vectors as an input to such algorithm can help the algorithm to "understand" the meaning of the entire document based on the semantic relations of its words. Since WordGraph2Vec is supposed to generate enhanced semantic vectors, document classification seems to be a natural task for this model. In our classification experiments, each document is represented by a vector calculated as the average of the embeddings of all document words (subject to the filtering specifications). The datasets that were used in this study are based on (Zhang et al., 2015) and they are listed in Table 3

Dataset	Class	train	test
AGs news	4	3000	1900
DBpedia	14	40000	5000
Yelp reviews	5	130000	10000
Yelp polarity	2	280000	19000
Yahoo	10	140000	5000
Amazon review	5	600000	130000
Amazon polarity	2	1800000	200000

Table 3: datasets information, columns are the number of classes in the dataset, number of samples in the training set and number of samples in the testing set, respectively.

Each document from the datasets was converted to a vector in the size of the embedding. For the classification task in this paper, we used the following three algorithms:

1. Neural Network algorithm:
   - Input layer size embedding size
   - three hidden layers of 256 neurons and dropout of 20%
   - dense layer
   - softmax activation
2. Random forest - 10 estimators, two min samples and no max depth
3. Logistic regression - L2 penalty, hinge loss and 1,000 iterations.

The parameters for Random Forest and Logistic regression are the default parameters in scikit-learn package in Python implementation. The Neural Network architecture was based on our previous experience in similar classification tasks.

We evaluated the proposed word embedding model in terms of model accuracy against the competing algorithms LINE, Word2Vec and Node2Vec described in section 5.1 .

5 Experiments

In this section, we present the settings of our experiments and their results.

algorithm	Mikolov test	Inflectional	Derivational	Encyclopedic	Lexicographic
Word2Vec	**71.0**	**77.9**	**31.8**	**25.7**	**15.2**
Node2Vec	53.2	59.2	7.5	20.4	14.3
Line-first	57.9	65.5	8.1	19.4	11.8
Line-second	49.3	75.6	13.1	15.4	13.7
Emb1	50.1	47.8	13.4	17.1	7.6
Emb2	60.8	68.9	21.4	20.6	10.7
Emb3	52.1	53.4	14.5	18.3	8.2
Emb4	63.3	71.9	24.7	21.9	11.6

Table 4: Accuracy percentage results of Mikolov Analogy test and the four main question types created in the Gladkova Analogy test.

5.1 Experimental Setups

We compared WordGraph2Vec with the following baselines:
Word2VEc: Window size 10, Skip-Gram method, embedding size 300.
Node2Vec: Window size 10. 80 nodes per walk. p and q equal to one. 10 rehearsals and embedding size 300.
LINE: Both first and second proximities. Embedding size of 300.
All baselines and WordGraph2Vec were trained on the Wikipedia dump described in section 4.1.
Using WordGraph2Vec we generated four different word embeddings, all were trained in window size 10, Skip-gram method and embedding size 300:
Emb1: Two context words (M = 2), one target word (T = 1) per context word. Two (N = 2) extra target words from the graph within the radius of one (R = 1).
Emb2: Two context words (M = 2), one target word (T = 1) per context word. One (N = 1) extra target word from the graph within the radius of one (R = 1).
Emb3: Three context words (M = 3), one target word (T = 1) per context word. Three (N = 3) extra target words from the graph within the radius of one (R = 1).
Emb4: Three context words (M = 3), one target word (T = 1) per context word. One (N = 1) extra target word from the graph within the radius of one (R = 1).

5.2 Experimental Results

5.2.1 Analogy test

We first compared the four tested embeddings of WordGraph2Vec against all baselines in terms of accuracy of the analogy test. In the Mikolov test, only 19,303 questions remained after we removed questions that contained words, which were filtered in advance. In the Gladkova test, only 89,484 questions remained after the same filtering operation as mentioned above. The four question types remained balanced.

As can be observed from Table 4, Word2Vec achieved the best results in both Mikolov and Gladkova analogy tests. Apparently, the low-level linear context is much more significant for analogy tasks in a natural language like English. However, Emb4 in WordGraph2Vec was ranked as the second best after Word2Vec. It may be no coincidence as Emb4 is the most similar to the Word2Vec implementation. Both Emb2 and Emb4 achieved better results than the competing graph-based algorithms. The Gladkova test, which is more balanced and does not contain significant outliers (such as country:city), present a more comprehensible result. With WordGraph2Vec, Emb2 and Emb4 achieved the highest accuracy scores in the Gladkova test, similar to the Mikolov's test. Only questions under Lexicographic type resulted in lower accuracy scores than the other baselines.

One reason, which might cause the lower accuracy results of WordGraph2Vec in comparison to Word2Vec, is the "noise" WordGraph2Vec creates. This "noise" can be described by the example below. Let us assume that there are two sentences in the corpus. The first, "University application could be tough". The second, "The safari in Kenya could be dangerous". With WordGraph2Vec, a valid word sequence could be "University application could be dangerous". The above sentence is less likely to appear in the language and might cause the word vectors of "university" and "dangerous" to be "closer" to each other though their

Algorithms	DBpedia	Ag News	Yelp	Yelp pol	Yahoo	Amazon	Amazon pol	Rank Std
Emb1	97.56	91.00	71.08	85.19	81.96	69.25	82.24	**1.40**
Emb2	97.62	91.12	71.36	85.61	82.05	69.44	82.70	1.56
Emb3	97.63	91.01	70.96	85.16	81.83	69.27	82.29	1.54
Emb4	97.65	91.04	70.79	85.75	81.98	**69.53**	82.78	2.13
LINE-first	97.55	90.98	**72.34**	83.87	82.43	69.45	81.07	2.25
LINE-second	97.52	91.14	71.15	85.81	82.22	69.11	82.54	2.01
Node2vec	97.14	90.91	70.69	82.80	81.84	68.81	79.19	3.35
word2vec	**97.69**	**91.31**	70.92	**86.48**	**82.44**	69.09	**83.08**	2.69

Table 5: Average results across the different classification algorithms. The values are the accuracy in percentage. Rank Std is the standard deviation values of the ranking of each embedding across all 21 combinations of classification algorithms and datasets.

semantic relationship is quite weak.

In further investigation of our results, we compared Word2Vec with Emb4 embedding, which is the closest to Word2Vec in terms of the training sentence set and Mikolov's analogy test results.

Out of 19,303 questions of the analogy test, 2,027 questions were answered correctly by Word2Vec and incorrectly by WordGraph2Vec. On the other hand, 515 questions were answered correctly by WordGraph2Vec only. The result for each question is a ranked list, which was calculated by the equation 3, of all the vocabulary words. Incorrect answer means the correct word is not in the top ten words on that list. Figure 3 presents a histogram of the positions of the correct words in the list mentioned above. The distribution can be described as a "long-tail" where in most cases (41.2%) the correct answer was ranked between positions 10-20. This could be because of the "noise" that was created by unrelated sentences. In addition, we examined the end of the tail as it is interesting to understand why in some questions the correct position drops from the top ten to more than 1000.

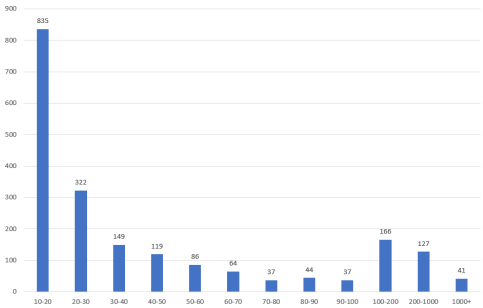

Figure 3: WordGraph2Vec histogram for the correct analogy results.

Figure 4 presents a table with Average and Median results for the amount of occurrences of a word c from equation 3 for each position group. The purpose of equation 3 is to find the best match for a word c that will be as close as possible to the subtraction of word vectors a and b. One theory that requires further investigation is presented in figure 4. WordGraph2Vec adds new sentences by choosing random words from each existing sentence and picking random words from a graph based on their occurrence frequency in the entire corpus. As the word c occurrence frequency in the corpus decreases, the results of the analogy tests containing this word drop. Presumably, this happens because a smaller amount of new training sentences contain the word c. Furthermore, we conducted a statistical comparison between the occurrence frequency of words in the ranking position 10-50 and 50+ and it was shown with $alpha$ = 0.001 that the average occurrence frequency is higher in the group of 10-50.

Group	Average	Median
10-20	89677	29876
20-30	77542	25494
30-40	59905	17154
40-50	63677	14253
50-60	59541	16025
60-100	47575	12318
100-500	47450	11439
500-1000	46421	5892
1000+	33596	5789

Figure 4: Word ranking positions vs. word frequencies.

5.2.2 Document classification

Each classification algorithm from section 5.1 was used to predict the document labels. The input was a vector representation of the document cre-

ated by the word embedding. In total 21 results were collected (seven datasets and three classification algorithm). As can be seen from Table 5, Word2Vec achieved the best average accuracy results on five datasets out of seven though the differences vs. most other algorithms do not appear significant. Emb2 was ranked in the second place in four datasets out of seven. Mostly, Word2Vec reached higher accuracy scores than Emb2, but the most interesting insight is that when Word2Vec achieved lower scores, Emb2 was still ranked in the second place, probably as a result of the new information gained from the graph.

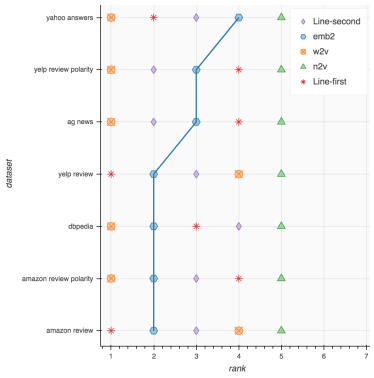

Figure 5: Accuracy ranking results for the best configuration of the proposed algorithm and the competing baselines. The x axis represents the average ranking of each embedding across the datasets.

In Figure 5, we see the ranking of Emb2 compared to all other baselines. Emb2 did not achieve the best accuracy scores but from Table 5 Emb2 had more stable results than the baselines as the standard deviation of its ranking is much lower than Line, Node2Vec and Word2Vec. From examining Figure 5 it seems that Line-2 is more stable but it has a higher standard deviation than Emb2 and its average ranking is lower. Node2Vec on average produced the lowest accuracy scores and it is inferior to the rest of the algorithms. Besides, Node2Vec is very unstable as its ranking has a standard deviation of 3.35, which is more than double than Emb2. In Figure 5, we can also see how the ranking of Line-1 varies from the first place to almost the last one. In order to verify the significance of the accuracy difference, we conducted a paired t-test. The test has shown that Emb2 has significantly higher accuracy results than the other baselines.

Table 6 presents the statistical significance results between all baselines and Emb2. For $p = 0.05$ the $t - value$ with 20 degrees of freedom is 2.086. Node2Vec was found significantly worse than Emb2. Word2Vec reached a higher accuracy score than Emb2, but the difference was not statistically significant.

baselines	d-avg	sd	se(d)	t
W2V	-0.16	0.55	0.001	-1.32
N2V	1.22	2.18	0.004	2.57
Line-1	0.32	1.17	0.002	1.25
Line-2	0.06	0.56	0.001	0.49

Table 6: Statistical significance results between the best configuration of the proposed algorithm, Emb2, and the baselines.

6 Conclusion

In this paper, we present WordGraph2Vec, a word embedding algorithm with semantic enhancement. The algorithm makes use of both linear and graph input in order to strengthen the semantic relations between words. Our experimental results show that the proposed embedding did not achieve the best results on analogy and classification tasks but was stable across the datasets and in most cases was ranked at the second place in terms of document classification and analogy tests accuracy. In future work, further settings of WordGraph2Vec can be explored, such as additional word graph configurations and a larger radius R for the new target words, which should yield target words that are not close to the context word in the original text. In addition, the proposed graph-based approach to word embedding can be evaluated on other NLP tasks in multiple languages.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–41. Association for Computational Linguistics.

Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering*, 19(3):355–369.

René Arnulfo García-Hernández, Yulia Ledeneva, Griselda Matías Mendoza, Ángel Hernández

Dominguez, Jorge Chavez, Alexander Gelbukh, and José Luis Tapia Fabela. 2009. Comparing commercial tools and state-of-the-art methods for generating text summaries. In *Artificial Intelligence, 2009. MICAI 2009. Eighth Mexican International Conference on*, pages 92–96. IEEE.

Jaume Gibert, Ernest Valveny, and Horst Bunke. 2012. Graph embedding in vector spaces by node attribute statistics. *Pattern Recognition*, 45(9):3072–3083.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*, pages 8–15.

Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM.

Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24. Association for Computational Linguistics.

Alex Markov, Mark Last, and Abraham Kandel. 2008. The hybrid representation model for web document classification. *International Journal of Intelligent Systems*, 23(6):654–679.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Adam Schenker, Horst Bunke, Mark Last, and Abraham Kandel. 2005. *Graph-Theoretic Techniques for Web Content Mining*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Adam Schenker, Mark Last, Horst Bunke, and Abraham Kandel. 2003. Classification of web documents using a graph model. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 240–244. IEEE.

Sheetal S Sonawane and Parag A Kulkarni. 2014. Graph based representation and analysis of text document: A survey of techniques. *International Journal of Computer Applications*, 96(19).

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Gladkova analogy test. a. *Gladkova analogy test*. Available at http://vecto.space/projects/BATS/.

Mikolov analogy test. b. *Mikolov analogy test*. Available at http://download.tensorflow.org/data/questions-words.txt.

Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community preserving network embedding. In *AAAI*, pages 203–209.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.