On Learning Word Embeddings From Linguistically Augmented Text Corpora

Amila Silva Chathurika Amarathunga
Singapore Management University
{amilasilva,chathurikaa}@smu.edu.sg

Abstract

Word embedding learning is a technique in Natural Language Processing (NLP) to map words into vector space representations, is one of the most popular research directions in modern NLP by virtue of its potential to boost the performance of many NLP downstream tasks. Nevertheless, most of the underlying word embedding methods such as word2vec and GloVe fail to produce high-quality representations if the text corpus is small and sparse. This paper proposes a method to generate effective word embeddings from limited data. Empirically, we show that the proposed model outperforms existing works for the classical word similarity task and for a domain-specific application.

1 Introduction

Representing words as feature vectors is a vital task in NLP. The trivial approach of representing words as distinct symbols is insufficient since it ignores semantic and syntactic similarities between words. As a result, distributional semantic models (DSMs) of word meanings (Clark, 2012; Erk, 2012) have been emerged, which were built on the hypothesis of *"words with similar meanings tend to appear in similar contexts"* (Harris, 1954). Most of the earliest DSMs for word representation learning are mainly based on clustering (Brown et al., 1992; Kneser and Ney, 1993; Uszkoreit and Brants, 2008) or factorizing (Bullinaria and Levy, 2007; Turney and Pantel, 2010; Baroni and Lenci, 2010; Ritter et al., 2010) global word co-occurrence matrix. However, with the introduction of neural word embedding methods by (Bengio et al., 2003), many studies (Collobert and Weston, 2008; Baroni et al., 2014) empirically prove that neural word embedding methods regularly and substantially outperform traditional DSMs. Thus, various neural models have been proposed recently for word representation learning.

Among the neural word embedding models, word2vec (Mikolov et al., 2013) is widely used in many NLP downstream tasks due to its efficiency in training and scalability. Word2vec learns word representations by maximizing the likelihood of the local context of words (defined using a window around a word). Following the light of word2vec, the variants of word2vec were introduced later with different context definitions. (Levy and Goldberg, 2014) introduced a model in which the contexts are defined by first-order dependency links between words. As extensions to the (Levy and Goldberg, 2014)'s work, (Komninos and Manandhar, 2016; Li et al., 2018) introduced second-order and higher-order dependency-based context for word embedding learning. Nevertheless, none of the existing neural models is capable to capture different types of contexts at once in their models. However, there are previous efforts (Minkov and Cohen, 2008) to design such a model using non-neural approaches.

Although the neural word embedding models have been proven useful in many NLP applications, the existing models have a few limitations. First, the existing works assume that the availability of large corpora, which may not be always available. Especially, the resources are limited to learn domain-specific embedding in most of the cases. Second, even though there are domain adaptation techniques (Bollegala et al., 2015) to overcome the scarcity of domain-specific resources in learning word embedding, it also requires a large amount of data from the source domain. Third, the existing works are only capable to capture one particular context, despite the fact that there are multiple ways to define context (Curran, 2004) using other linguistic relations (i.e., using dependency relations, using co-reference relations).

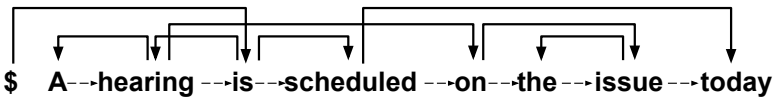


Figure 1: A sentence as a network, each word is a node in the network and edges are obtained from dependency links (solid links) and linear links (dashed links).

To overcome these limitations, here we propose a method to expand the number of sentences in a small text corpus by linguistically generating multiple versions for each sentence. To do so, a network is constructed for each sentence such that different types of context appeared within the neighbourhood of each word in the sentence. Then the multiple versions of the sentence are generated by exploring the linguistic network using a biased random walk approach (discussed in detail in Section 2), motivated by previous network representation learning techniques (Perozzi et al., 2014; Grover and Leskovec, 2016).

The rest of the paper is structured as follows. In Section 2, the technical details of our approach is presented. In Section 3, we evaluate our model using two different tasks and assess the hyperparameter sensitivity of our approach. We conclude the manuscript with some promising directions for future works in Section 4.

2 Proposed Approach

Let us first discuss the word2vec skip-gram model (Mikolov et al., 2013). For a given set of sentences, the skip-gram model loops on the words in each sentence and tries to use the current word to predict its neighbors. Formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, the skip-gram model maximizes the following equation,

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\geq j\leq c} log\ p(w_{t+j}|w_t) \tag{1}$$

where c is the window size around the center word. In word2vec, each word-context pair acts as a data point in training. However, if the given corpus is small, it produces only a smaller number of word-context pairs, which might not be enough to train a word2vec model. The intuition behind our model is to generate multiple versions of the same sentence, which captures different contexts in each version and consequently provide more word-context pairs to learn the skip-gram model.

2.1 Random Walks Over Sentences

The contexts defined by a window approach (*linear-based*) and dependency relations (*dependency-based*) are considered within the scope of this paper to construct the linguistic network of a sentence as shown in Figure 1, in which nodes are the words in the sentence and dependency links and linear links create the edges. However, our model can be easily scaled to capture the contexts defined using other linguistic structures such as POS tag and named entity sequences.

We perform random walks of fixed length starting from the first word of each sentence. The formal procedure of random walks is described as follows: Let n_i and n_{i-1} denote the i^{th} and $i-1^{th}$ nodes in the walk and n_i is sampled from the following probability distribution:

$$P(n_i = x|n_{i-1} = y) = \begin{cases} \frac{\pi_{yx}}{Z} & \text{If } (y,x)\ \epsilon\ E \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where π_{yx} is the unnormalized transition probability between the nodes y and x, Z is the normalizing factor ($Z = \sum_{\forall\,(y,z)\,\epsilon\,E}\ \pi_{yz}$), and E is the edge list. The transition probabilities (see Section 2.2) between nodes are set in a manner that the multiple random walks of a sentence have different local-contexts in their words. Then the random walks generated for all the sentences are used to train a word2vec model as illustrated in Figure 2.

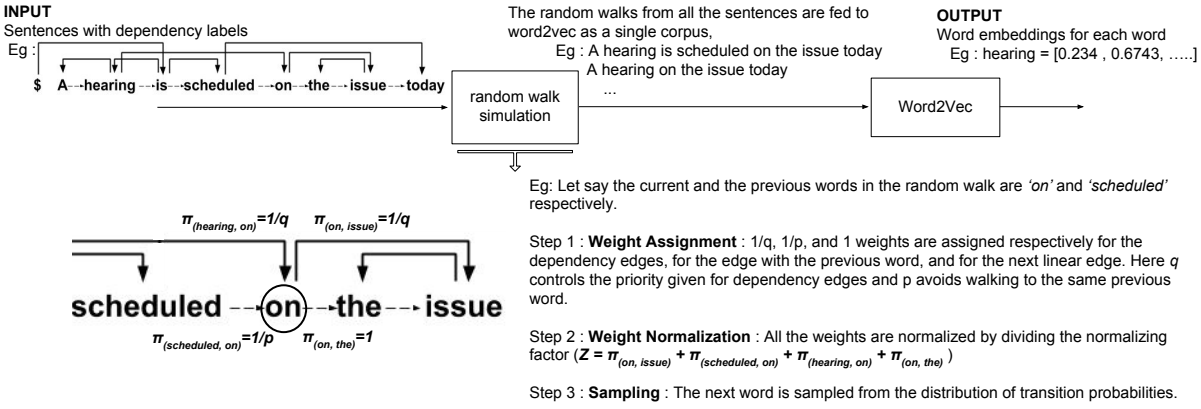


Figure 2: The proposed model for learning word embeddings

2.2 Biasing For Different Contexts

To give different priorities for different edge types, each edge is weighted (w_{yx}) based on the edge type and the nodes are sampled based on the edge weights $(\pi_{yx} = w_{yx})$ as follows; Let $x,\ y,\ z$ denote the previous, current, and next words in a random walk respectively.

$$\pi_{yx} = \begin{cases} 1/p & \text{if } x = z \\ 1 & \text{if } (y,z)\ \epsilon\ E_{linear} \\ 1/q & \text{if } (y,z)\ \epsilon\ E_{dependency} \end{cases} \tag{3}$$

where parameters p and q are the hyperparameters to bias the random walk procedure and $E_{dependency}$ and E_{linear} represent the lists of edges based on dependency and linear links respectively.

By setting a high value for p $(> max(1, q))$, we can avoid the immediate reappearance of same words in random walks. The parameter q controls the priority that is given for the dependency links. If q is high (> 1), priority will be given for the linear edges. In contrast, if q is low (< 1), priority will be given for the dependency based edges. Hence, two extreme cases will be able to capture *linear-based* and *dependency-based* context respectively.

3 Experiments

3.1 Dataset Construction

For the word similarity task, the embeddings were trained on a smaller section of English Wikipedia corpus (Al-Rfou et al., 2013)[1], which contains 1,911,951 sentences, 52,468,613 tokens and 555,688 unique words. In addition, we trained malware-domain specific embeddings using a corpus extracted from APTnotes, a repository of publicly-available papers and blogs related to malicious campaigns, activity, and software [2]. We have chosen 193 reports from the year 2010 to 2015. Since APTnotes are in PDF format, PDFMiner tool (Shinyama, 2004) has been used to convert PDF files into plain text format. After removing the non-sentences, headers, and footers; this malware-related text corpus consists of 27,553 sentences, 108,311 tokens, and 37,857 unique words. Spacy[3] is used to obtain dependency labels for both datasets.

3.2 Baseline Methods

We consider following baseline methods to assess the effectiveness of our approach,

[1]https://sites.google.com/site/rmyeid/projects/polyglot
[2]https://github.com/aptnotes/data
[3]https://spacy.io/

Table 1: Results on word similarity/relatedness.

	SVD	W2V	GloVe	Ext	W2V | Ext	GloVe | Ext	Our Model
SIMLEX	0.2611	0.2828	0.2413	0.1531	0.2306	0.2530	**0.2991**
WS-353	0.6055	0.6098	0.5230	0.4532	0.6266	0.5554	**0.6616**
MEN	0.5232	0.5078	0.5799	0.4681	0.5651	0.6069	**0.6293**

- **SVD** (Bullinaria and Levy, 2007): SVD decomposition of the PPMI matrix (Only top 10000 frequent words are considered to generate PPMI matrix due to the computational complexity).
- **word2vec** (Mikolov et al., 2013): The original skip-gram model based on negative sampling.
- **GloVe** (Pennington et al., 2014): This model efficiently leverages global statistical information through factorizing a word-word co-occurrence matrix.
- **Ext** (Komninos and Manandhar, 2016): This model uses dependency based context for learning word embeddings. It introduces second-order dependency into the model proposed by (Levy and Goldberg, 2014).

In addition, the concatenations of linear-context baselines and dependency-context baselines are also considered as a separate set of baselines. For a fair comparison, we set dimension of the embeddings to 300, the number of negative samples to 5, and the context window size to 10 in all the baselines. More details about the experimental setups are discussed in Section 3.3.

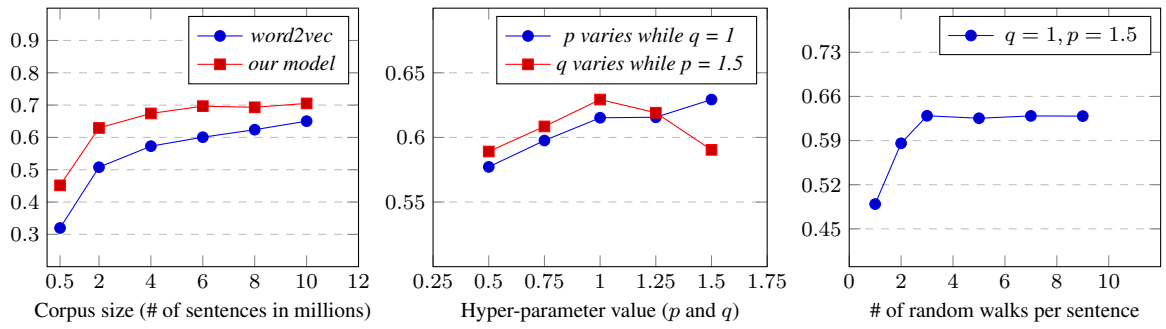
3.3 Results

3.3.1 Word Similarity/Relatedness Task

This is a widely used task to evaluate the effectiveness of word embeddings. We use three different datasets, namely: (a) SIMLEX-999 (Hill et al., 2015), (b) WS-353 (Agirre et al., 2009), and (c) MEN (Bruni et al., 2014). Each dataset uses a different notion of word similarity for scoring. The cosine value between two word vectors is used to measure the degree of similarity/relatedness between them. The Spearmans rank correlation coefficient (Myers et al., 2013) is used to check the correlation of ranks between the human annotation and computed similarities. The hyperparameters of the models are tuned using the development set of the MEN dataset. Consequently, 1.5, 1, and 3 are set to p, q, and $number\ of\ walks$ hyperparameters respectively. The same set of hyperparameter values is used for all the experiments.

According to the results in Table 1, our model outperforms the baselines considerably for all the datasets. Even though the traditional distributional models like SVD have been proven effective for small corpora in some cases, SVD doesn't perform well here. The concatenation of *linear-based* and *dependency-based* baselines shows improvements over their individual models, which further shows the importance of capturing different types of context together in learning embeddings.

Parameter Sensitivity: As shown in Figure 3a, our model considerably outperforms word2vec when the corpus size is small. However, the improvement against word2vec are declined when the corpus grows. This observation clearly supports the fact that our model is especially effective for small corpora. Figure 3b shows the results for different p and q values. In our model, the extreme ends of the scale of q represents the merely *linear-based* ($q >> 1$) and merely *dependency-based* ($q << 1$) systems. Hence, the good performance with the setting of $q = 1$ means that the capturing of both the linear-based and dependency-based contexts together leads to good performance. With this setting, target words of the sentence can access to its local context and dependency context and also to the local context of the dependency related words. This is where our model is superior compared to the computationally expensive method, which enumerate all the possible dependency and linear paths separately.As shown in Figure 3c, we observe that increasing the *number of walks per sentence* improves the performance until the parameter reaches 3. The performance remains consistent for further increments of the parameter.



(a) (b) (c)

Figure 3: Results (using MEN dataset) for word similarity/relatedness task: (a) for different sizes of the corpora, (b) for different p and q settings, and (c) for different number of random walks per sentence. y axes represent the Spearman's rank correlation coefficient.

Table 2: Results on malware-domain specific sentence classification.

	SVD	W2V	GloVe	Ext	W2V | Ext	GloVe | Ext	Cross-Domain	Our Model
CNN-Static	80.98	81.23	81.94	81.55	80.96	82.43	80.73	**83.12**
CNN-Non-static	81.12	81.42	82.74	82.17	81.45	83.19	80.84	**83.55**

This shows that the use of multiple random walks per sentence leads to capture more useful context of the words, though it is getting saturated after some point.

3.3.2 Malware Related Sentence Classification

In this task, pretrained embeddings are used along with a Convolutional Neural Network (CNN) (Kim, 2014) to perform malware domain-sentence classification task (introduced in SubTask 1 of SemEval-2018 Task 8 - SecureNLP (Phandi et al., 2018)). 10-fold cross-validation is used for the evaluation. Since the domain adaptation techniques have been using to overcome the data constraints in learning domain-specific word embeddings, we consider simple cross-domain word embedding learning baseline in this experiment as an addition. In this baseline, source (Wikipedia corpus) and target (malware corpus) domains are combined to form a single corpus to learn word embeddings.

As shown in Table 2, our approach outperforms other baselines for both CNN-static (embeddings are kept static in training) and CNN-non-static (embeddings are further tuned in training) versions of (Kim, 2014)'s model. Surprisingly, the addition of data from another domain doesn't lead to good performance. Especially, the improvement is significant for CNN-static version, which further shows the quality of the embeddings generated by our model. In addition, the good performance for CNN-non-static version shows that our embeddings are useful for initialization purposes, as pretrained embeddings are only used to initialize word embeddings in the CNN-non-static model.

4 Conclusion

In this paper, we propose a model to capture both *linear* and *dependency-based* contexts together in learning word embeddings. Our model outperforms the well-known baselines for the word similarity task and domain-specific sentence classification task.

Although our model is empirically effective, different types of context (i.e., *dependency-based* and *linear-based*) might not be useful for all the words in learning their representation. Hence more sophisticated walking approach to walk on linguistic structures is worth exploring. Moreover, the combination of other linguistic structures (i.e., co-reference, NER, and POS tag sequences) to the proposed model might be another promising research direction.

References

Agirre, E., E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proc. of HLT-NAACL*.

Al-Rfou, R., B. Perozzi, and S. Skiena (2013). Polyglot: Distributed word representations for multilingual nlp. In *Proc. of CoNLL*.

Baroni, M., G. Dinu, and G. Kruszewski (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proc. of ACL*.

Baroni, M. and A. Lenci (2010). Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*.

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin (2003). A neural probabilistic language model. *Journal of machine learning research*.

Bollegala, D., T. Maehara, and K. ichi Kawarabayashi (2015). Unsupervised cross-domain word representation learning. In *Proc. of ACL*.

Brown, P. F., P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai (1992). Class-based n-gram models of natural language. *Computational linguistics*.

Bruni, E., N. Tram, M. Baroni, et al. (2014). Multimodal distributional semantics. *The Journal of Artificial Intelligence Research*.

Bullinaria, J. A. and J. P. Levy (2007). Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*.

Clark, S. (2012). Vector space models of lexical meaning. *Handbook of Contemporary Semantics–second edition. Wiley-Blackwell*.

Collobert, R. and J. Weston (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*.

Curran, J. R. (2004). From distributional to semantic similarity.

Erk, K. (2012). Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*.

Grover, A. and J. Leskovec (2016). node2vec: Scalable feature learning for networks. In *Proc. of ACM SIGKDD*.

Harris, Z. S. (1954). Distributional structure. *Word*.

Hill, F., R. Reichart, and A. Korhonen (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Journal of Computational Linguistics*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proc. of EMNLP*.

Kneser, R. and H. Ney (1993). Improved clustering techniques for class-based statistical language modelling. In *Proc. of EUROSPEECH*.

Komninos, A. and S. Manandhar (2016). Dependency based embeddings for sentence classification tasks. In *Proc. of HLT-NAACL*.

Levy, O. and Y. Goldberg (2014). Dependency-based word embeddings. In *Proc. of ACL*.

Li, C., J. Li, Y. Song, and Z. Lin (2018). Training and evaluating improved dependency-based word embeddings.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*.

Minkov, E. and W. W. Cohen (2008). Learning graph walk based similarity measures for parsed text. In *Proc. of EMNLP*.

Myers, J. L., A. D. Well, and R. F. Lorch Jr (2013). *Research design and statistical analysis*. Routledge.

Pennington, J., R. Socher, and C. Manning (2014). Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Perozzi, B., R. Al-Rfou, and S. Skiena (2014). Deepwalk: Online learning of social representations. In *Proc. of ACM SIGKDD*.

Phandi, P., A. Silva, and W. Lu (2018). Semeval-2018 Task 8: Semantic Extraction from CybersecUrity REports using Natural Language Processing (SecureNLP). In *Proc. of SemEval*.

Ritter, A., O. Etzioni, et al. (2010). A latent dirichlet allocation method for selectional preferences. In *Proc. of ACL*.

Shinyama, Y. (2004). PDFMiner. `https:/Interscience/euske.github.io/pdfminer/`.

Turney, P. D. and P. Pantel (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*.

Uszkoreit, J. and T. Brants (2008). Distributed word clustering for large scale class-based language modeling in machine translation. *Proc. of ACL-08: HLT*.