Graph-based Syntactic Word Embeddings

Ragheb Al-Ghezi
Aalto University
`ragheb.al-ghezi@aalto.fi`

Mikko Kurimo
Aalto University
`mikko.kurimo@aalto.fi`

Abstract

We propose a simple and efficient framework to learn syntactic embeddings based on information derived from constituency parse trees. Using biased random walk methods, our embeddings not only encode syntactic information about words, but they also capture contextual information. We also propose a method to train the embeddings on multiple constituency parse trees to ensure the encoding of global syntactic representation. Quantitative evaluation of the embeddings shows competitive performance on POS tagging task when compared to other types of embeddings, and qualitative evaluation reveals interesting facts about the syntactic typology learned by these embeddings.

1 Introduction

Distributional similarity methods have been the standard learning representation in NLP. Word representations methods such as Word2vec, GloVe, and FastText [1, 2, 3] aim to create vector representation to words from other words or characters that mutually appear in the same context. The underlying premise is that ”a word can be defined by its company” [4]. For example, in the sentences, ”I eat an apple every day” and ”I eat an orange every day”, the words ’orange’ and ’apple’ are similar as they share similar contexts.

Recent approaches have proposed a syntax-based extension to distributional word embeddings to include functional similarity in the word vectors by leveraging the power of dependency parsing[5] [6]. Syntactic word embeddings have been shown to be advantageous in specific NLP tasks such as question type classification[7], semantic role labeling[8], part-of-speech tagging[6], biomedical event trigger identification[9], and predicting brain activation patterns [10]. One limitation of these methods is that they do not encode the hierarchical syntactic structure of which a word is a part due to its reliance on non-constituency parsing such as dependency parsing. While the latter analyzes the grammatical structure of a sentence by establishing a directed binary head-dependent relation among its words, constituency parsing analyzes the syntactic structure of a sentence according to a phrase structure grammar.

Syntactic hierarchy has advantages in tasks such as grammar checking, question answering, and information extraction [11]. It has also been encoded in neural models such as Recursive Neural Tensor Network and has proved it can predict the compositional semantic effects of sentiment in language [12]. Moreover, it can uniquely disambiguate the functional role of some words and therefore the overall semantic meaning. Figure 1 shows the modal verb *should* in the following sentences: *(1) Let me know should you have any question.* and *(2) I should study harder for the next exam.* Even though the word *should* is a modal verb (MD) in both sentences, it exhibits two different grammatical functions: conditionality and necessity respectively. Similarly, the word *is* in *(3) The king is at home.* and *(4) Is the king at home?* has a similar semantic meaning in both sentences, yet it exhibits two different syntactic roles (statement-forming and question-forming). Traditional word embeddings methods give a contextual, semantic representation to words like *is* and *should*, but they make no distinction of their grammatical function due to the absence of information on syntactic hierarchy. On the other hand, constituency parse trees provide a syntactic representation that can easily capture such distinction. Figures (a) and (b) show constituency parse tree of sentences (1) and (2) respectively. The difference in the position of the modal verb *should* in both sentences indicates a difference in the grammatical function, especially if it is compared to words with similar grammatical function in other sentences such as the one in (figure (c)). Comparing figures (a) and (c), we can note that *should* hold the same sense of conditionality the word *if* has. To this end, we propose a simple, graph-based framework to build syntactic word embeddings that can be flexibly customized to capture syntactic as well as contextual information by leveraging information derived from either manually or automatically constituency-parsed trees.

While recent transformer-based models such as BERT [13] have proved to be more sophisticated than word embeddings, the latter remains a popular choice due to its simplicity and efficiency. Thus, the contribution of this work is two-fold: (1) bridge the research gap in the literature of word embedding by introducing hierarchical syntactic embeddings based on constituency parsing (2) propose a graph-theoretic training method that cluster words according to their syntactic and constituent role without sacrificing the original context in which a word appears.

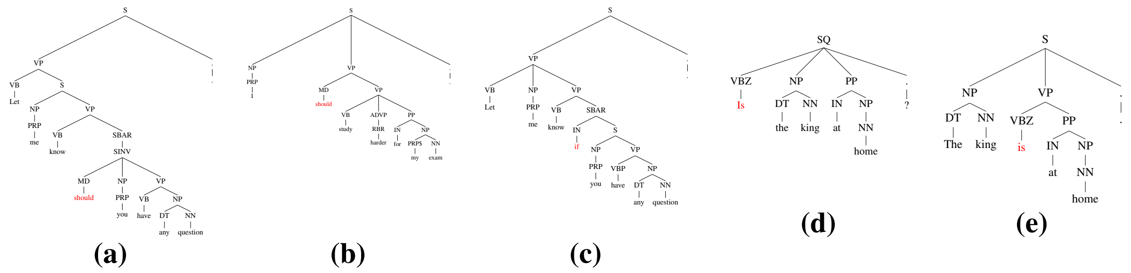

Figure 1: Trees (a), (b), (d) and (e) show different positions of the words *should* and *is* respectively indicating differences in syntactic functions. Tree (c) is analogous to (a) suggesting similarity in syntactic function between *should* and *if*.

2 Related Work

The NLP literature is rich with studies suggesting an improvement to original word embeddings models by incorporating external semantic resources like lexicons and ontologies [14, 15, 16, 17, 18]. However, very few studies were dedicated to syntactic embeddings. One of the earliest methods was dependency-based word embeddings [5], which generalizes the Skip-gram algorithm to include arbitrary word context. Instead of using bag-of-word context, they use context derived automatically from dependency parse trees. Specifically, for a word w with modifiers $m_1, ..., m_k$ and head h, the contexts $(m1, lbl_1), ..., (m_k, lbl_k), (h, lbl_h^1)$, where *lbl* is a type of dependency relation between the head. and the modifier (e.g. nsub, dobj, etc). For example, the context for the word *scientist* in *"Australian scientist discovers star with telescope"* is Australian/amod and discovers/$nsubj^1$.

Another modification to *word2vec* model was proposed by [6] to improve the word embeddings to syntax-based tasks by making it sensitive to the positioning of the words, and thereby accounting for its lack of order-dependence. The modification does not involve incorporating external parsing information, but it includes using 2 output predictors for every word in the window context each of which is dedicated to predicting position-specific value. Results on syntax-based tasks such as POS tagging and parsing show an improvement over classic word2vec embeddings.

More recently, a new approach, named SynGCN, for learning dependency-based syntactic embeddings is introduced by [19]. SynGCN builds syntactic word representation by using Graph Convolution Network (GCN). Using GCN allows SynGCN to capture global information from the graph on which it was trained while remaining efficient at training due to parallelization. Experiments show that SynGCN obtains improvement over state-of-the-art approaches when used with methods such as ELMo [20].

Most syntactic word embeddings methods rely on dependency parsing, and to the best of our knowledge that our work is the first utilizing constituency parsing to build syntactic representation.

3 Method

Our goal is to learn word embeddings that not only capture the sentence-level syntactic hierarchy encoded by the constituency parse tree, but also capture a global (suprasentential) syntactic representation, and because the constituency parse tree only provides sentence-level syntactic representations, we need a method to combine multiple constituency parse trees. We also need a flexible algorithm to learn the embeddings from those combined trees. In this section, we present a method of parse tree combination (namely graph unionization) as well as the Node2vec algorithm.

Graph Unionization Given a training dataset of constituency parse trees, we compose one graph (henceforth supergraph, Figure 2) by unionizing all the sentence trees in the training dataset. Formally, let $G(V, E)$ be a graph in the training corpus, where V represents a lexical or a non-lexical vertex in a constituency parse tree and E is the edge between them, and let H be $\bigcup_{i=1}^{n} G_i(V_i, E_i)$ where $\bigcup$ is a non-disjoint union operator and n is the number of sentences in the training corpus. The vertices and edges of the supergraph V_H and E_H are $\bigcup_{i=1}^{n} V_i$ and $\bigcup_{i=1}^{n} E_i$ respectively [21, 22].

Node2vec For learning syntactic embeddings from the supergraph, we use a variant of skip-gram algorithm, called node2vec algorithm[23]. Node2vec adapts Word2vec algorithm to graphs in which a node is defined by an arbitrary set of other nodes in the same graph sampled using a biased random walk. Using tunable parameters p and q, the biased random walk offers BFS and DFS search behavior in which more diverse neighborhoods are explored, and therefore richer representation may be learned[23].

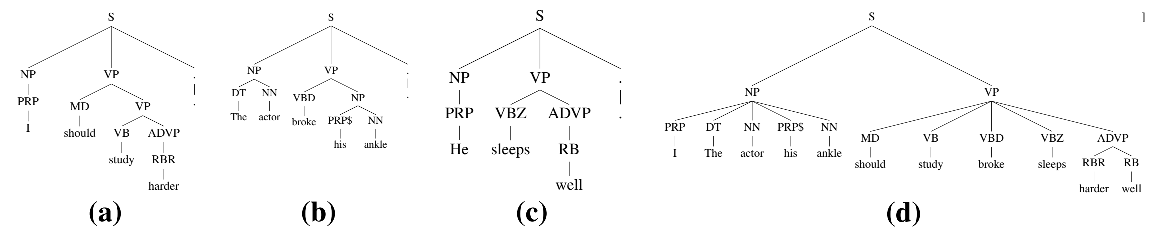

Figure 2: Trees (a), (b) and (c) are sample parse trees in the training corpus while (d) represents their unionized supergraph.

As shown in Figure 2, nodes tagged with certain labels, such as adjectives (JJ) or nouns (NN), will be linked together in the supergraph while remaining children of a noun phrase (NP). Similarly, sentences with similar grammatical structures such as interrogative sentences or questions (SQ) will be clustered together in the supergraph. It can be noted that the supergraph can cluster words of similar syntactic functions together while simultaneously enforcing/preserving the global syntactic hierarchy of the training corpus. The supergraph with the aid of the biased sampling strategy Node2vec offers the flexibility of learning customizable syntactic representation. A breadth-first search strategy, for instance, would favor the selection of words of similar POS tags and thereby yielding word-class-specific representation while a depth-first strategy would yield more hierarchical or contextual representation.

4 Data and Experiment

For the purposes of training the syntactic embeddings, we use the Penn Treebank corpus[24], which contains over 43,000 constituency parse trees to sentences collected from the Wall Street Journal (WSJ). Next, we unionize all the parses trees into one supergraph. The supergraph has 51071 vertices and 65895 edges, and it has an average degree of 2.5805 and a density of 5.05×10^{-5}. We chose to unionize all the trees in the training corpus for simplicity, but we certainly could have grouped the sentences into clusters of thematic or semantic identity prior to applying unionization. After that, we train the embeddings with node2vec algorithm using SGD of 10,000 epochs and a learning rate of 0.025 with a weight decay of 0.005. In terms of node2vec hyperparameters, we chose a random walk of length 200 and a batch size of 100, and the return parameter p and the in-out parameter q are both set to $10^{-}6$. Lower p values keep the walk close to the starting node, and lower q values encourage the walk to behave in a DFS manner [23]. The training took 51 seconds on 1 Tesla K80 GPU using the Graphvite Python package[25]. We initialize the word vectors randomly for simplicity, but initialization with other types of distributional embeddings such as word2vec or GloVe is possible. We decide to explore the latter in future work.

5 Results

We conduct two types of evaluations: qualitative and quantitative. In the qualitative evaluation, we examine the extent to which the learned embeddings can encode grammatical information about the words using words analogies and word arithmetics. In addition, we compare its performance against GloVe vectors [2] and SynGCN [19] on one stream task, POS tagging.

5.1 Qualitative Evaluation

One common method to evaluate word embeddings is examining word vectors by their top k nearest neighbors in the latent vector space. From table 5.1, we observe that the top 3 neighbors for the words *complicate*, *failed*, *earthquakes* are all of the similar syntactic category: a present verb attracts similar present verbs; a plural noun attracts plural nouns; and so on. Similarly, the adjective *responsible* and the adverb *handsomely* maintain a very close distance to words of the same part-of-speech. In contrast, neither GloVe vectors nor SynGCN exhibit similar neighborhood typology. This confirms that our constituency-based embeddings have consistently preserved syntactic information about words.

Another way to evaluate word embeddings is by explaining word analogies by the means of word vector arithmetics [26]. The famous example used in [1] is *woman is to queen as man is to king*, or $(w_q + w_k) - w_m \approx w_w$. When we apply the same method to our constituency-based syntactic vectors, we assert that the vector arithmetic sense strongly matches the syntactic analogies. For example, in table 5.1, if we subtract the sum of word vectors in the prepositional phrase (PP) *in an industrial* from the PP *of any clearly domestic*, the top 3 nearest neighbors in our embeddings are all adverbs (ADV) to compensate for the missing adverb in the second PP. We also note the case is not true for the other types of embeddings where the top nearest neighbors are affected by words in the PP. Similarly, applying the same arithmetic operations to the phrases *his state-of-the-art plan* and *her plan* would results in adjective vectors, unlike the other embeddings.

word	**SynGCN (Cos Sim)**		**GloVe (Cos Sim)**		**Ours (Cos Sim)**	
complicate	complicates	0.60	complicating	0.90	**prune**	0.97
	complicating	0.51	complicates	0.87	**ruin**	0.97
	simplify	0.43	**jeopardize**	0.83	**outpace**	0.97
failed	failing	0.61	failing	0.89	**reconstructed**	0.98
	unsuccessful	0.56	attempt	0.88	**slated**	0.98
	fail	0.54	attempts	0.86	**challenged**	0.98
earthquakes	earthquake	0.65	**quakes**	0.90	**failures**	0.98
	tsunamis	0.61	**aftershocks**	0.80	**sons**	0.98
	quakes	0.58	**tremors**	0.79	**bouts**	0.98
handsomely	**generously**	0.48	doled	0.71	**effectively**	0.97
	ornately	0.46	rewarded	0.70	**negligently**	0.97
	competently	0.45	gambled	0.68	**inexorably**	0.97
responsible	**accountable**	0.52	**involved**	0.90	**extensive**	0.97
	responsibility	0.41	responsibility	0.84	**fifth-biggest**	0.97
	tasked	0.39	planning	0.80	**one-woman**	0.97
([of]+[any]+[clearly]+[domestic]) - ([in] + [an] + [industrial])	none	0.28	ignore	0.76	**precariously**	0.88
	all	0.27	acknowledge	0.76	**slightly**	0.87
	interestingly	0.26	**necessarily**	0.75	**equitably**	0.87
([his]+[state-of-the-art]+[plan]) - ([her] + [assignment])	plans	0.35	build	0.69	**government-held**	0.92
	cutting-edge	0.34	renovation	0.66	**harder-line**	0.90
	planning	0.34	redevelopment	0.65	**front-page**	0.90
([would] + [need])-[require]	**will**	0.52	we	0.91	**will**	0.92
	should	0.59	come	0.90	**might**	0.91
	could	0.47	want	0.88	**can**	0.91

Table 1: Comparison of top 3 KNN with cosine similarity produced by SynGCN [19], GloVe [2], and our embeddings. Words in bold belong to the same POS tag/ grammatical category.

5.2 Intrinsic Evaluation

We also test the performance of our constituency-based embeddings on a mainstream task, parts-of-speech tagging. Our goal is not to achieve state-of-the-art results in POS tagging, but we want to demonstrate the grammatical potential of our embeddings. For this purpose, we treat POS tagging as

	SVM- F1 Score	CRF- F1 Score
Glove [2]	0.731	0.894
SynGCN [19]	**0.892**	0.898
Ours	0.881	**0.910**

Table 2: Evaluation on POS tagging using SVM and CRF classifiers. Scores represent a mean F1 score of 5-fold cross-validation.

an independent classification task (as opposed to structured prediction one) in which a non-sequential classifier support vector machine (SVM) is used to predict a POS tag for a word acontextually. The use of non-neural, non-sequential classifier ensures that the grammatical generalizability comes strictly from the embeddings and not from the neural network or the context. Nevertheless, we also treat POS tagging as a structured prediction task in which we use a sequential classifier like conditional random field (CRF) for the purposes of comparison. Performance is also reported for two other word embeddings: GloVe and SynGCN under the same settings.

We test the performance using the trained vectors on the first 2000 sentences of the Brown corpus [27]. In table 2, we report the mean F1 score of 5-fold cross-validation in which we can observe that our vectors are competitive in performance to SynGCN and far better than GloVe when used with SVM classifier. In addition, our embeddings outperform both of the competing embeddings when used with CRF.

Even though the performance of the constituency-based embeddings slightly lags behind SynGCN in the case of independent classification, the size of the corpus upon which our embeddings were trained (Penn Treebanks 1 million tokens) is much smaller compared to the one upon which SynGCN was trained (Wikipedia 1.1 billion tokens). In addition, the flexibility of learning customizable syntactic word embeddings as well as the training efficiency make constituency-based word embeddings a powerful and promising research direction that can be applied to other graph-based tasks.

6 Conclusion and Future Work

We presented a simple and efficient framework to learn syntactic embeddings from constituency parse trees using a combination of multiple graph unionization and biased random walk. Our framework can be flexibly customized to learn purely contextual and non-contextual syntactic embeddings, and it can be also used as a post-hoc method for other kinds of (distributional) word embeddings. Thus, for future studies, we would like to investigate training constituency-based vectors on a larger corpus and examine the effect of different initialization on more mainstream tasks such as machine translation and automatic speech recognition.

7 Acknowledgments

We would like to thank Sandeep Suntwal, Yaroslav Getman, and Katja Voskoboinik for their helpful discussion and insightful comments. We also thank the anonymous reviewers for their constructive comments. This work was funded by the Academy of Finland (grant number 322625).

References

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

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

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors

with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[4] J.R. Firth and F.R. Palmer. *Selected Papers of J.R. Firth, 1952-1959*. Indiana University Studies in the History and Theory of Linguistics. Longmans, 1968.

[5] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, 2014.

[6] Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, 2015.

[7] Alexandros Komninos and Suresh Manandhar. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1490–1500, 2016.

[8] Michael Roth and Mirella Lapata. Neural semantic role labeling with dependency path embeddings. *arXiv preprint arXiv:1605.07515*, 2016.

[9] Jian Wang, Jianhai Zhang, Yuan An, Hongfei Lin, Zhihao Yang, Yijia Zhang, and Yuanyuan Sun. Biomedical event trigger detection by dependency-based word embedding. *BMC medical genomics*, 9(2):45, 2016.

[10] Samira Abnar, Rasyan Ahmed, Max Mijnheer, and Willem Zuidema. Experiential, distributional and dependency-based word embeddings have complementary roles in decoding brain activity. *arXiv preprint arXiv:1711.09285*, 2017.

[11] Dan Jurafsky and James H Martin. Speech and language processing. vol. 3, 2014.

[12] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

[15] Douwe Kiela, Felix Hill, and Stephen Clark. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048, 2015.

[16] Julien Tissier, Christophe Gravier, and Amaury Habrard. Dict2vec: Learning word embeddings using lexical dictionaries. 2017.

[17] Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 1219–1228, 2014.

[18] Tom Bosc and Pascal Vincent. Auto-encoding dictionary definitions into consistent word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1532, 2018.

[19] Shikhar Vashishth, Manik Bhandari, Prateek Yadav, Piyush Rai, Chiranjib Bhattacharyya, and Partha Talukdar. Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. *arXiv preprint arXiv:1809.04283*, 2018.

[20] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[21] Frank Harary. *Graph Theory. Addison Wesley Publishing Company.* 1969.

[22] Jonathan L Gross and Jay Yellen. *Graph theory and its applications*. CRC press, 2005.

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

[24] Mitchell P Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. Treebank-3. *Linguistic Data Consortium, Philadelphia*, 14, 1999.

[25] Zhaocheng Zhu, Shizhen Xu, Meng Qu, and Jian Tang. Graphvite: A high-performance cpu-gpu hybrid system for node embedding. In *The World Wide Web Conference*, pages 2494–2504. ACM, 2019.

[26] Carl Allen and Timothy Hospedales. Analogies explained: Towards understanding word embeddings. *arXiv preprint arXiv:1901.09813*, 2019.

[27] W Nelson Francis and Henry Kucera. Brown corpus manual. *Letters to the Editor*, 5(2):7, 1979.