Two/Too Simple Adaptations of Word2Vec for Syntax Problems

Conference Paper · May 2015

DOI: 10.3115/v1/N15-1142

4 authors, including:

Wang Ling
Carnegie Mellon University
26 PUBLICATIONS **165** CITATIONS
SEE PROFILE

Isabel Trancoso
Inesc-ID / IST, Lisbon, Portugal
288 PUBLICATIONS **1,862** CITATIONS
SEE PROFILE

Two/Too Simple Adaptations of Word2Vec for Syntax Problems

Wang Ling **Chris Dyer** **Alan Black** **Isabel Trancoso**
L^{2}F Spoken Systems Lab, INESC-ID, Lisbon, Portugal
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
Instituto Superior Técnico, Lisbon, Portugal
{lingwang,cdyer,awb}@cs.cmu.edu
isabel.trancoso@inesc-id.pt

Abstract

We present two simple modifications to the models in the popular Word2Vec tool, in order to generate embeddings more suited to tasks involving syntax. The main issue with the original models is the fact that they are insensitive to word order. While order independence is useful for inducing semantic representations, this leads to suboptimal results when they are used to solve syntax-based problems. We show improvements in part-of-speech tagging and dependency parsing using our proposed models.

1 Introduction

Word representations learned from neural language models have been shown to improve many NLP tasks, such as part-of-speech tagging (Collobert et al., 2011), dependency parsing (Chen and Manning, 2014; Kong et al., 2014) and machine translation (Liu et al., 2014; Kalchbrenner and Blunsom, 2013; Devlin et al., 2014; Sutskever et al., 2014). These low-dimensional representations are learned as parameters in a language model and trained to maximize the likelihood of a large corpus of raw text. They are then incorporated as features along side hand-engineered features (Turian et al., 2010), or used to initialize the parameters of neural networks targeting tasks for which substantially less training data is available (Hinton and Salakhutdinov, 2012; Erhan et al., 2010; Guo et al., 2014).

One of the most widely used tools for building word vectors are the models described in (Mikolov et al., 2013), implemented in the Word2Vec tool, in particular the "skip-gram" and the "continuous bag-of-words" (CBOW) models. These two models make different independence and conditioning assumptions; however, both models discard word order information in how they account for context. Thus, embeddings built using these models have been shown to capture semantic information between words, and pre-training using these models has been shown to lead to major improvements in many tasks (Collobert et al., 2011). While more sophisticated approaches have been proposed (Dhillon et al., 2011; Huang et al., 2012; Faruqui and Dyer, 2014; Levy and Goldberg, 2014; Yang and Eisenstein, 2015), Word2Vec remains a popular choice due to their efficiency and simplicity.

However, as these models are insensitive to word order, embeddings built using these models are suboptimal for tasks involving syntax, such as part-of-speech tagging or dependency parsing. This is because syntax defines "what words go where?", while semantics than "what words go together". Obviously, in a model where word order is discarded, the many syntactic relations between words cannot be captured properly. For instance, while most words occur with the word *the*, only nouns tend to occur exactly afterwords (e.g. *the cat*). This is supported by empirical evidence that suggests that order-insensitivity does indeed lead to substandard syntactic representations (Andreas and Klein, 2014; Bansal et al., 2014), where systems using pre-trained with Word2Vec models yield slight improvements while the computationally far more expensive which use word order information embeddings of Collobert et al. (2011) yielded much better results.

In this work, we describe two simple modifications to `Word2Vec`, one for the skip-gram model and one for the CBOW model, that improve the quality of the embeddings for syntax-based tasks[1]. Our goal is to improve the final embeddings while maintaining the simplicity and efficiency of the original models. We demonstrate the effectiveness of our approaches by training, on commodity hardware, on datasets containing more than 50 million sentences and over 1 billion words in less than a day, and show that our methods lead to improvements when used in state-of-the-art neural network systems for part-of-speech tagging and dependency parsing, relative to the original models.

2 Word2Vec

The work in (Mikolov et al., 2013) is a popular choice for pre-training the projection matrix $W \in \Re^{d \times |V|}$ where d is the embedding dimension with the vocabulary V. As an unsupervised task that is trained on raw text, it builds word embeddings by maximizing the likelihood that words are predicted from their context or vice versa. Two models were defined, the skip-gram model and the continuous bag-of-words model, illustrated in Figure 1.

The skip-gram model's objective function is to maximize the likelihood of the prediction of contextual words given the center word. More formally, given a document of T words, we wish to maximize

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

Where c is a hyperparameter defining the window of context words. To obtain the output probability $p(w_o|w_i)$, the model estimates a matrix $O \in \Re^{|V| \times d_w}$, which maps the embeddings r_{w_i} into a $|V|$-dimensional vector o_{w_i}. Then, the probability of predicting the word w_o given the word w_i is defined as:

$$p(w_o \mid w_i) = \frac{e^{o_{w_i}(w_o)}}{\sum_{w \in V} e^{o_{w_i}(w)}} \qquad (2)$$

This is referred as the softmax objective. However, for larger vocabularies it is inefficient to compute o_{w_i}, since this requires the computation of a $|V| \times d_w$ matrix multiplication. Solutions for problem are addressed in the `Word2Vec` by using the hierarchical softmax objective function or resorting to negative sampling (Goldberg and Levy, 2014).

The CBOW model predicts the center word w_o given a representation of the surrounding words $w_{-c}, ..., w_{-1}, w_1, w_c$. Thus, the output vector $o_{w_{-c},...,w_{-1},w_1,w_c}$ is obtained from the product of the matrix $O \in \Re^{|V| \times d_w}$ with the sum of the embeddings of the context words $\sum_{-c \leq j \leq c, j \neq 0} r_{w_j}$.

We can observe that in both methods, the *order of the context words does not influence the prediction output*. As such, while these methods may find similar representations for semantically similar words, they are less likely to representations based on the syntactic properties of the words.

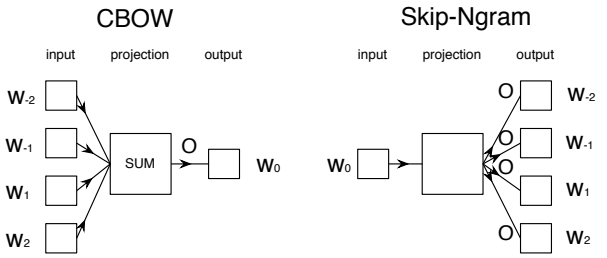

Figure 1: Illustration of the Skip-gram and Continuous Bag-of-Word (CBOW) models.

3 Structured Word2Vec

To account for the lack of order-dependence in the above models, we propose two simple modifications to these methods that include ordering information, which we expect will lead to more syntactically-oriented embeddings. These models are illustrated in Figure 2.

3.1 Structured Skip-gram Model

The skip-gram model uses a single output matrix $O \in \Re^{|V| \times d}$ to predict every contextual word $w_{-c}, ..., w_{-1}, w_1, ..., w_c$, given the embeddings of the center word w_0. Our approach adapts the model so that it is sensitive to the positioning of the words. It defines a set of $c \times 2$ output predictors $O_{-c}, ..., O_{-1}, O_1, O_c$, with size $O \in \Re^{(|V|) \times d}$. Each of the output matrixes is dedicated to predicting the

[1]The code developed in this work is made available in https://github.com/wlin12/wang2vec.

output for a specific relative position to the center word. When making a prediction $p(w_o \mid w_i)$, we select the appropriate output matrix O_{o-i} to project the word embeddings to the output vector. Note, that the number of operations that must be performed for the forward and backward passes in the network remains the same, as we are simply switching the output layer O for each different word index.

3.2 Continuous Window Model

The Continuous Bag-Of-Words words model defines a window of words $w_{-c}, ..., w_c$ with size c, where the prediction of the center word w_0 is conditioned on the remaining words $w_{-c}, ..., w_{-1}, w_1, ..., w_c$. The prediction matrix $O \in \Re^{(|V|)\times d}$ is fed with the sum of the embeddings of the context words. As such, the order of the contextual words does not influence the prediction of the center word. Our approach defines a different output predictor $O \in \Re^{(|V|\times 2cd}$ which receives as input a $(2c \times d)$-dimensional vector that is the concatenation of the embeddings of the context words in the order they occur $[e(w_{-c}), \ldots, e(w_{-1}), e(w_1), \ldots, e(w_c)]$. As matrix O defines a parameter for the word embeddings for each relative position, this allows the words to be treated differently depending on where they occur. This model, denoted as CWindow, is essentially the window-based model described in (Collobert et al., 2011), with the exception that we do not project the vector of word embeddings into a window embedding before making the final prediction.

In both models, we are increasing the number of parameters of matrix O by a factor of $c \times 2$, which can lead to sparcity problems when training on small datasets. However, these models are generally trained on datasets in the order of 100 millions of words, where these issues are not as severe.

4 Experiments

We conducted experiments in two mainstream syntax-based tasks part-of-speech Tagging and Dependency parsing. Part-of-speech tagging is a word labeling task, where each word is to be labelled with its corresponding part-of-speech. In dependency parsing, the goal is to predict a tree built of syntactic relations between words. In both tasks, it has been shown that pre-trained embeddings can be used to achieve better generalization (Collobert et al., 2011; Chen and Manning, 2014).

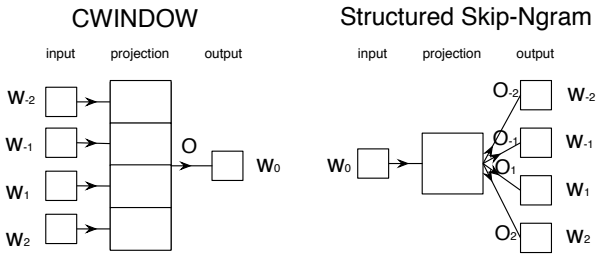

Figure 2: Illustration of the Structured Skip-gram and Continuous Window (CWindow) models.

4.1 Building Word Vectors

We built vectors for English in two very different domains. Firstly, we used an English Wikipedia dump containing 1,897 million words (60 million sentences), collected in September of 2014. We built word embeddings using the original and our proposed methods on this dataset. These embeddings will be denoted as *WIKI(L)*. Then, we took a sample of 56 million English tweets with 847 million words collected in (Owoputi et al., 2013), and applied the same procedure to build the *TWITTER* embeddings. Finally, we also use the Wikipedia documents, with 16 million words, provided in the `Word2Vec` package for contrastive purposes, denoted as *WIKI(S)*. As preprocessing, the text was lowercased and groups of contiguous digits were replaced by a special word. For all corpora, we trained the network with a $c = 5$, with a negative sampling value of 10 and filter out words with less than 40 instances. WIKI(L) and TWITTER have a vocabulary of 424,882 and 216,871 types, respectively, and embeddings of 50 dimensions.

Table 1 shows the similarity for a few selected keywords for each of the different embeddings. We can see hints that our models tend to group words that are more syntactically related. In fact, for the word *breaking*, the CWindow model's top five words are exclusively composed by verbs in the continuous form, while the Structured Skip-gram model tends to combine these with other forms of the verb break. The original models tend to be less keen on

Embeddings	WIKI(S)	TWITTER	WIKI(L)
query	*breaking*	*amazing*	*person*
CBOW	breaks	incredible	someone
	turning	awesome	anyone
	broke	fantastic	oneself
	break	phenomenal	woman
	stumbled	awsome	if
Skip-gram	break	incredible	harasser
	breaks	awesome	themself
	broke	fantastic	declarant
	down	phenominal	someone
	broken	phenomenal	right-thinking
CWindow (this work)	putting	incredible	woman
	turning	amaaazing	man
	sticking	awesome	child
	pulling	amzing	grandparent
	picking	a-mazing	servicemember
Structured Skip-gram (this work)	break	incredible	declarant
	turning	awesome	circumstance
	putting	amaaazing	woman
	out	ah-mazing	schoolchild
	breaks	amzing	someone

Table 1: Most similar words using different word embedding models for the words *breaking*, *amazing* and *person*. Each word is queried in a different dataset.

preserving such properties. As for the TWITTER embeddings, we can observe that our adapted embeddings are much better at finding lexical variations of the words, such as *a-mazing*, resembling the results obtained using brown clusters (Owoputi et al., 2013). Finally, for the query *person*, we can see that our models tend to associate this term to other words in the same class, such as *man*, *woman* and *child*, while original models tend to include unrelated words, such as *if* and *right-thinking*.

In terms of computation speed, the Skip-gram and CBOW models, achieve a processing rate of 71.35k and 342.17k words per second, respectively. The Structured Skip-gram and CWindow models can process 34.64k and 124.43k words per second, respectively. There is a large drop in computational speed in the CWindow model compared to the CBOW model, as it uses a larger output matrix, which grows with the size of the window. The Structured Skip-gram model processes words at almost half the speed of the Skip-gram model. This is explained by the fact that the Skip-gram model subsamples context words, varying the size of the window size stochastically, so that words closer to the center word are sampled more frequently. That is, when defining a window size of 5, the actual window size used for each sample is a random value between 1 and 5. As we use a separate output layer for each position, we did not find this property to be useful as it provides less training samples for output matrixes with higher indexes. While our models are slower they are still suitable for processing large datasets as all the embeddings we use were all built within a day.

4.2 Part-Of-Speech Tagging

We reimplemented the window-based model proposed in (Collobert et al., 2011), which defines a 3-layer perceptron. In this network, words are first projected into embeddings, which are concatenated and projected into a window embedding. These are finally projected into an output layer with size of the POS tag vocabulary, followed by a softmax. In our experiments, we used a window size of 5, word embeddings of size 50 and window embeddings of size 500. Word embeddings were initialized using the pre-trained vectors and these parameters are updated as the rest of the network. Additionally, we also add a capitalization feature which indicates whether the first letter of the work is uppercased, as all word features are lowercased words. Finally, for words unseen in the training set and in the pre-trained embeddings, we replace them with a special unknown token, which is also modelled as a word type with a set of 50 parameters. At training time, we stochastically replace word types that only occur once in the training dataset with the unknown token. Evaluation is performed with the part-of-speech tag accuracy, which denotes the percentage of words labelled correctly.

Experiments are performed on two datasets, the English Penn Treebank (PTB) dataset using the standard train, dev and test splits, and the ARK dataset (Gimpel et al., 2011), with 1000 training, 327 dev and 500 labelled English tweets from Twitter. For the PTB dataset, we use the WIKI(L) embeddings and use TWITTER embeddings for the ARK dataset. Finally, the set of parameters with the highest accuracy in the dev set are used to report the score for the test set.

Results are shown in Table 2, where we observe that our adapted models tend to yield better re-

	PTB		Twitter	
	Dev	Test	Dev	Test
CBOW	95.89	96.13	87.85	87.54
Skip-gram	96.62	96.68	88.84	88.73
CWindow	**96.99**	97.01	**89.72**	89.63
Structured Skip-gram	96.62	**97.05**	89.69	**89.79**
SENNA	96.54	96.58	84.96	84.85

Table 2: Results for part-of-speech tagging using different word embeddings (rows) on different datasets (PTB and Twitter). Cells indicate the part-of-speech accuracy of each experiment.

	Dev		Test	
	UAS	LAS	UAS	LAS
CBOW	91.74	88.74	91.52	88.93
Skip-gram	92.12	89.30	91.90	89.55
CWindow	92.38	89.62	92.00	89.70
Structured Skip-gram	**92.49**	**89.78**	**92.24**	**89.92**
SENNA	92.24	89.30	92.03	89.51

Table 3: Results for dependency parsing on PTB using different word embeddings (rows). Columns UAS and LAS indicate the labelled attachment score and the unlabelled parsing scores, respectively.

sults than the original models in both datasets. In the Twitter dataset, our results slightly higher than the accuracy reported using only Brown clusters in (Owoputi et al., 2013), which was 89.50. We also try initializing our embeddings with those in (Collobert et al., 2011), which are in the "Senna" row. Even though results are higher in our models, we cannot conclude that our method is better as they are trained crawls from Wikipedia in different time periods. However, it is a good reference to show that our embeddings are on par with those learned using more sophisticated models.

4.3 Dependency Parsing

The evaluation on dependency parsing is performed on the English PTB, with the standard train, dev and test splits with Stanford Dependencies. We use neural network defined in (Chen and Manning, 2014), with the default hyper-parameters[2] and trained for 5000 iterations. The word projections are initialized using WIKI(L) embeddings. Evaluation is performed with the labelled (LAS) and unlabeled (UAS) attachment scores.

In Table 3, we can observe that results are consistent with those in part-of-speech tagging, where our models obtain higher scores than the original models and with competitive results compared to Senna embeddings. This suggests that our models are suited at learning syntactic relations between words.

5 Conclusions

In this work, we present two modifications to the original models in Word2Vec that improve the word embeddings obtained for syntactically motivated tasks. This is done by introducing changes that make the network aware of the relative positioning of context words. With these models we obtain improvements in two mainstream NLP tasks, namely part-of-speech tagging and dependency parsing, and results generalize in both clean and noisy domains.

[2]Found in http://nlp.stanford.edu/software/nndep.shtml

Acknowledgements

This work was partially supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds, and also through projects CMU-PT/HuMach/0039/2008 and CMU-PT/0005/2007. The PhD thesis of Wang Ling is supported by FCT grant SFRH/BD/51157/2010. The authors also wish to thank the anonymous reviewers for many helpful comments.

References

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax. In *Proceedings of ACL*.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014.

Fast and robust neural network joint models for statistical machine translation. In *52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, June*.

Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. 2011. Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems*, pages 199–207.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of EMNLP*.

Geoffrey E Hinton and Ruslan Salakhutdinov. 2012. A better way to pretrain deep boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 2447–2455.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*, pages 1700–1709.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proc. of EMNLP*, pages 1001–1012, Doha, Qatar, October.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proceedings of ACL*, pages 1491–1500.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *HLT-NAACL*, pages 380–390.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.