Dependency Based Embeddings for Sentence Classification Tasks

Alexandros Komninos
Department of Computer Science
University of York
UK
ak1153@york.ac.uk

Suresh Manandhar
Department of Computer Science
University of York
UK
suresh@cs.york.ac.uk

Draft 14 March 2016. To appear in NAACL HLT 2016.

Abstract

We compare different word embeddings from a standard window based skipgram model, a skipgram model trained using dependency context features and a novel skipgram variant that utilizes additional information from dependency graphs. We explore the effectiveness of the different types of word embeddings for word similarity and sentence classification tasks. We consider three common sentence classification tasks: question type classification on the TREC dataset, binary sentiment classification on Stanford's Sentiment Treebank and semantic relation classification on SemEval 2010 dataset. For each task we use three different classification methods: a Support Vector Machine, a Convolutional Neural Network and a Long Short Term Memory Network. Our experiments show that dependency based embeddings can outperform standard window based embeddings in most of the tasks while using dependency context embeddings as additional features improves performance in all tasks regardless of the classification method.

1 Introduction

Representing words as low dimensional vectors (also known as word embeddings) has been a widely adopted technique in NLP. Word representations can be used as features for classification tasks such as named entity recognition or chunking (Turian et al., 2010), and as a pretraining method for initializing deep neural network representations (Collobert et al., 2011; Kim, 2014). Word embeddings provide better generalization to unseen examples since they can capture general semantic and syntactic properties of words. One of the most popular methods of learning word embeddings is the skipgram model of Mikolov et al. (2013a; 2013b) where embeddings are trained by making predictions of context words appearing in a window around a target word.

The standard skipgram model ignores syntax and only partially takes into consideration the sequential structure of text, but still captures certain syntactic properties of words. A significant amount of previous research has explored methods for directly taking syntax into account for word embedding learning (Baroni et al., 2015; Cheng and Kartsaklis, 2015; Hashimoto et al., 2014). One simple method is based on traditional count-based distributional semantic spaces and utilizes words with syntactic types from a dependency parse graph as context features (Padó and Lapata, 2007; Baroni and Lenci, 2010). This method has also been applied to skipgram models, where words predict dependency context features instead of other words (Levy and Goldberg, 2014).

Syntax based embeddings have been shown to have different properties in word similarity evaluations than their window based counterparts, better capturing the functional properties of words. However, it is not clear if they provide any advantage for NLP tasks. We show that using dependency context features can be a general method of providing syntactic information for several sentence classification tasks . Furthermore, the dependency context embeddings improve performance with all classifiers we tested.

We consider the usage of word and dependency context features for three common sentence classification tasks: TREC question type classification, binary sentiment prediction on Stanford Sentiment Treebank, and SemEval 2010 relation identification. We evaluate different methods of using the dependency context embeddings as extra features besides word embeddings to inject information about the syntactic structure of a sentence. The advantage of such a method is that it can be applied to any classifier that utilizes standard word embeddings. We evaluate the usefulness of syntax based word embeddings and dependency context embeddings with three different sentence classification methods: a Support Vector Machine (SVM), a Convolutional Neural Network (CNN) and a Long Short Term Memory network (LSTM).

In order to better utilize the structure of dependency graphs, we propose an extended version of the simple dependency based skipgram of Levy et al. (2014). This extended version considers co-occurrences in a dependency graph between pairs of words, words and dependency context features, and between different dependency context features. This scheme results in word embeddings that share properties between window based models and dependency graph based ones. More importantly, it provides additional structural information for the dependency context feature embeddings making them more effective when used in sentence classification tasks.

Our evaluation provides several insights on the role of syntax for embeddings and how they can be used for sentence classification. First, we confirm past claims about the different properties between dependency and window based skipgram embeddings in word similarity tasks. Second, we show that dependency based embeddings perform better in question classification and relation identification than window based ones. These results are robust across multiple classification methods. We show that combining dependency context feature embeddings together with word embeddings provide a simple and effective way to improve sentence classification performance. Finally, the performance gain is higher for the extended dependency based skipgram developed in this paper.

2 Related Work

Estimating word representations from text has been the focus of a lot of research in NLP. Traditional count-based models learn representations by applying SVD in a word-word co-occurrence matrix (Turney et al., 2010). More recently, neural models have been used to learn word embeddings by optimizing for a word prediction task (Collobert et al., 2011; Mnih and Teh, 2012; Mikolov et al., 2013a). However, the most commonly used word representation techniques like word2vec's skipgram and CBoW take little consideration of syntactic structure.

Several modifications have been proposed so that word embedding learning algorithms can better utilize syntax or the sequence structure of sentences. The dependency based skipgram model of (Levy and Goldberg, 2014) is one of the models we consider for experimentation in this paper and further extend it. Evaluation of such models is limited to word similarity or lexical substitution in context (Melamud et al., 2015), and little is known about performance within other NLP tasks. Hashimoto et al. (2014) train a similar model based on a log-bilinear language model and predicate-argument structures and report improvements on phrase similarity tasks compared to standard skipgram. In Ling et al. (2015), skipgram and CBoW models are adapted to include position specific weights for the words inside the co-occurrence window and the resulting embeddings provide slight improvements for parsing and POS tagging tasks. The C-PHRASE model (Baroni et al., 2015) is another modification of the CBoW model that uses an external parser to replace windows with syntactic constituents. In Cheng and Kartsaklis (2015), a recursive neural network structured according to a sentence's parse learns word embeddings by composing into valid sentences rather than distorted ones.

Our work is also related to methods of providing explicit syntactic information to sentence classifiers. Most of the previous work rely on tree-structured neural architectures to drive composition of word embeddings to a sentence representation (Socher et al., 2012; Tai et al., 2015; Li et al., 2015). We use a different approach where syntactic information is provided only through embeddings. Our approach is not orthogonal to using tree-structured models, but has the advantage that it can use large amounts of automatically parsed text data to learn about dependency types.

3 Embedding Models

The skipgram model of Mikolov et al.(2013a; 2013b) optimize vector representations of words (word embeddings) such that they can predict other context words occurring in a small window. The architecture consists of a single hidden layer feedforward network without any non-linearity. The input of the network is the index of a target word (a one-hot vector) and the output is a vector of probabilities of appearance for context words. The network learns word embeddings by maximizing the log probability of a context word c given a target word t observed in a large corpus of textual data D:

$$\sum_{t,c \in D} \log p(c|t) \tag{1}$$

To avoid the large computational cost of applying a softmax for the whole vocabulary, a commonly used strategy is to train with negative sampling. For each target-context pair (t, c) coming from the observed data D, a small number of context words is sampled from unobserved data $D^{'}$ according to a simple distribution and used as the negative classes. The objective becomes:

$$\arg\max_{v_t, v_c} \sum_{t,c \in D} \log \sigma(v_t \cdot v_c) + \sum_{t,c \in D'} \log \sigma(-v_t \cdot v_c) \tag{2}$$

where v_t and v_c are target and context word embeddings, and σ is the sigmoid function.

The network learns two sets of weights for each word: one for embedding words to a low dimensional representation in the hidden layer that we will refer to as the embedding layer weights, and one for assigning a probability to context words that we will refer to as the prediction layer weights. Both sets of weights assign representations to words such that words that have similar co-occurrence patterns with other words are closer in the embedding space. Typically, the embedding layer weights are used as feature representations of words for other other tasks. Due to its scalability to large corpora and the good performance of its derived word embeddings in several NLP tasks the skipgram model has become a standard solution for unsupervised learning of word representations.

While typical training of skipgarm is performed by optimizing for the prediction of other words in a window around the target word, it is possible to use other contextual features, such as contexts from dependency graphs of sentences.

We consider three variations of skipgram based on different target-context pairs:

3.1 Window-5 based skipgram (Win5)

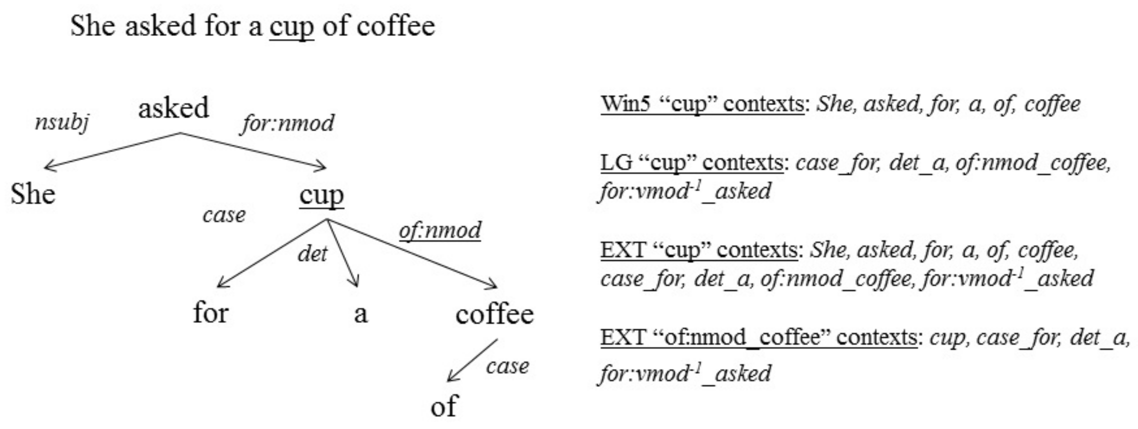
This is a standard skipgram model that considers target-context word pairs inside a window of 5 words to the right and to the left of the target word. The window size for every target instance in the corpus is uniformly sampled from the [1,5] range, effectively providing a weighting scheme for context words according to their distance from the target word.

3.2 Skipgram with dependency contexts (LG)

Levy and Golberg's (2014) modification to the skipgram model replaces context words in a window by dependency contexts. A dependency context is a discrete symbol denoting a word and its syntactic role in a dependency parse graph (e.g. $nsubj\_she$, $of : nmod\_coffee$, $of : nmod^{-1}\_cup$). The directionality of dependency edges is encoded by introducing features with inverse relations. Training of this skipgram variant is similar to window based approaches, but each word is considered as a node in a dependency graph obtained by a parser, and embeddings are optimized to predict their corresponding word's immediate syntactic contexts. The network's weight matrices have different shapes, where representations coming from the embedding layer weights correspond to word embeddings, while representations coming from the prediction layer weights to dependency context embeddings.

3.3 Extended Skipgram (EXT)

We propose another variation of skipgram based on dependency graphs that utilizes additional co-occurrences compared to the LG variant. Each target word is taken as a node in the dependency graph and then optimize embeddings such that they maximize the probability of other words within distance one and two in the graph. As with the Win5 model, we



Win5 "cup" contexts: *She, asked, for, a, of, coffee*

LG "cup" contexts: *case_for, det_a, of:nmod_coffee, for:vmod^{-1}_asked*

EXT "cup" contexts: *She, asked, for, a, of, coffee, case_for, det_a, of:nmod_coffee, for:vmod^{-1}_asked*

EXT "of:nmod_coffee" contexts: *cup, case_for, det_a, for:vmod^{-1}_asked*

Figure 1: A sentence and its dependency parse graph. The contexts of the word "cup" are shown for each model. In addition, for the EXT model the contexts of the "of:nmod_coffee" dependency context feature are shown.

apply a weighting according to distance, with words having distance one from the target counted twice. This word-word prediction behaves similarly to the Win5 model, but considers the dependency parse to filter coincidental co-occurrences. The second type of predictions that embeddings are optimized for is similar to the LG model, where each word predicts its dependency contexts. We also optimize for a third type of context prediction where for each node, dependency contexts become the targets and predict the rest of dependency contexts of the same node. An example of the different target-context pairs that each skipgarm variant utilizes can be seen in Figure 1. The three types of target-context pairs for the extended skipgram are interleaved during training. The weight matrices of this network are symmetric resulting in two embeddings per word and dependency context feature.

3.4 Implementation Details

We trained 300 dimensional versions of the above skipgram variants on English Wikipedia August 2015 dump of 2 billion words. Vocabularies consist of words and dependency contexts that appear more than 100 times (approximately 220k words and 1.3m dependency contexts). Training was done by applying negative sampling with 15 negative samples per target-context pair for 10 iterations over the entire corpus using stochastic gradient descent. The following typical methods were applied during training: drawing negative samples according to their unigram distribution raised to the power of 0.75, linear decay of learning rate with initial $\alpha = 0.25$, and subsampling of target words with probability given by $p = \frac{f-10^{-5}}{f} - \sqrt{\frac{10^{-5}}{t}}$ where f is the word's frequency. Dependency parsing for LG and EXT training was done with the Stanford Neural Network dependency parser (Chen and Manning, 2014) using Universal Dependency tags (De Marneffe et al., 2014).

4 Word Similarity Evaluation

We evaluate the effect of the different contextual features for skipgram word embeddings in two word similarity datasets: WordSim-353 (Finkelstein et al., 2001) and SimLex-999 (Hill et al., 2014). For both datasets, we compare the cosine similarity of word embeddings for a pair of words to human judgements and report Spearman's correlation on Table 1. The two datasets use a different notion of word similarity for scoring. Wordsim-353 mostly captures topical similarity (or relatedness), giving high similarity to pair of words like clothes-closet. LexSim-999 uses a more strict version of similarity, often called substitutional similarity, where the pair clothes-closet has a low similarity score. Win5 skipgram version achieves a higher correlation for WordSim-353 compared to LG, but the results are reversed for SimLex-999. This agrees with previous research that shows that syntactic contexts correlate better with substitutional similarity judgements than using words in a window as contexts (Levy and Goldberg, 2014). As expected, the extended model represents a middle ground solution between the two. While similarity based evaluation makes obvious that different contextual features capture different properties of words, it is not clear which kind similarity notion is more useful when word representations are used as features for NLP tasks. We answer this question for sentence level classification tasks in the next section.

Embeddings	**WordSim-353**	**SimLex-999**
Win5	**0.714**	0.389
LG	0.621	**0.460**
EXT	0.678	0.414

Table 1: Spearman correlation for the 3 skipgram variants on WordSim-353 and SimLex-999 word similarity evaluation tasks.

5 Sentence Classification

We consider three common sentence classification tasks: TREC question type classification (QC), binary sentiment classification on Stanford's Sentiment Treebank (SST), and relation identification in the SemEval 2010 dataset. The experiments aim to answer two questions. First, to assess the effect of different context features for word embeddings when used sentence classification tasks, given their different behaviour on word similarity evaluation. Second, to experiment with methods of using the dependency context embeddings themselves as a way to provide classifiers with dependency syntactic information. We carry experiments with three different classification methods: SVMs on averaged embeddings, the Convolutional Neural Network of Kim (2014), and a Long Short Term Memory recurrent neural network (Hochreiter and Schmidhuber, 1997). These classifiers have some distinct characteristics. The BoE SVM does not take into account the structure of the sentence, nor does it build any internal representations. On the other hand, both the CNN and LSTM networks operate on sequences of words and build internal representations before predicting the class label distribution. However, they do not have access to explicit syntactic information.

We first give a description of the classification methods and the way embeddings are used as features, followed by the description of the tasks and results.

5.1 Classification Methods

SVM with averaged embeddings We create a sentence representationby averaging embeddings of sentence features (words and dependency contexts). This can be considered the equivalent of a Bag-of-Words sentence representation in the embedding space, hence called Bag-of-Embeddings (BoE). We then train a classifier by applying an SVM with a Gaussian kernel:

$$K(\mathbf{x}, \mathbf{x}^{'}) = \exp(-\gamma \|\mathbf{x} - \mathbf{x}^{'}\|^2) \tag{3}$$

For hyperparameter tuning, we set parameter γ of the kernel to $1/k$, where k is the number of features (dimensionality of embeddings), and then perform cross validation for the c parameter using the standard Win5 word embeddings in the question classification task.

Convolutional Neural Network (CNN) We use the simple Convolutional Neural Network of Kim (2014) that has been shown to perform well in multiple sentence classification tasks. The network's input is a sentence matrix X formed by concatenating k-dimensional word embeddings. Then a convolutional filter $W \in \mathbb{R}^{h \times k}$ is applied to every possible sequence of length h to get a feature map:

$$c_i = tanh(W \cdot X + b) \tag{4}$$

followed by a max-over-time pooling operation to get the feature with the highest value:

$$\hat{c} = \max \mathbf{c} \tag{5}$$

The pooled features of different filters are then concatenated and passed to a fully connected softmax layer to perform the classification. The network uses multiple filters with different sequence sizes covering different size of windows in the sentence. All hyperparameters of the network are the same as used in the original paper (Kim, 2014): stochastic dropout (Srivastava et al., 2014) with $p = 0.5$ on the penultimate layer, 100 filters for each filter region with filter regions of width 2,3 and 4. Optimization is performed with Adadelta (Zeiler, 2012) on mini-batches of size 50.

Long Short Term Memory (LSTM) LSTM networks (Hochreiter and Schmidhuber, 1997) are recurrent neural networks where recurrent units consist of a memory cell c and three gates i, o and f. Given a sequence of input embeddings $\mathbf{x}$, LSTM outputs a sequence of states $\mathbf{h}$ given by the following equations:

$$\begin{pmatrix} \mathbf{i_t} \\ \mathbf{f_t} \\ \mathbf{o_t} \\ \tilde{\mathbf{c_t}} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \cdot \begin{pmatrix} \mathbf{h_{t-1}} \\ \mathbf{x_t} \end{pmatrix} \tag{6}$$

$$\mathbf{c_t} = \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \tilde{\mathbf{c_t}} \tag{7}$$

$$\mathbf{h_t} = \mathbf{o_t} \odot tanh(\mathbf{c_t}) \tag{8}$$

where $W \in \mathbb{R}^{4k \times 2k}$, $\tilde{\mathbf{c_t}}$ is a candidate state for the memory cell and $\odot$ is element-wise vector multiplication. The distribution of labels for the whole sentence is computed by a fully connected softmax layer on top of the final hidden state after applying stochastic dropout with $p = 0.25$. We use 150 dimensions for the size of $\mathbf{h}$, Adagrad (Duchi et al., 2011) for optimization and mini-batch size of 100.

We provide syntactic information to each classifier in the following manner. First we parse each sentence to get a dependency graph. Each node in the graph is associated with a word w having an embedding $\mathbf{v_w}$ and a set of dependency context features $d_1, d_2, ..., d_C$ with embeddings $\mathbf{v_{d_1}}, \mathbf{v_{d_2}}, ..., \mathbf{v_{d_C}}$ exactly like during the dependency based skipgram training process. We then create a representation $\mathbf{x}$ of that node using different combinations of its associated word and dependency context embeddings:

- *Words*: Using only word embeddings

$$\mathbf{x} = \mathbf{v_w} \tag{9}$$

- *Dep*: A node's representation becomes the average of its associated dependency context embeddings:

$$\mathbf{x} = \frac{1}{C} \sum_{c=1}^{C} \mathbf{v_{d_c}} \tag{10}$$

- *Wavg*:Combination of the word and dependency context embeddings by a weighted average scheme that assigns equal contribution to the word and dependency context part:

$$\mathbf{x} = \frac{1}{2}\mathbf{v_w} + \frac{1}{2C} \sum_{c=1}^{C} \mathbf{v_{d_c}} \tag{11}$$

- *Conc*: Similar to the Wavg, but dependency context embeddings are first averaged and then concatenated to the word embedding to form a single vector:

$$\mathbf{x} = \mathbf{v_w} \oplus \frac{1}{C} \sum_{c=1}^{C} \mathbf{v_{d_c}} \tag{12}$$

  where $\oplus$ is the concatenation operator. This method keeps the word and syntactic part separate at the expense of doubling the dimensionality.

The above methods are used with the LG and EXT variants to create context specific node representations. For the EXT model, both word and dependency context embeddings used come from the embedding layer weights. The *Words* method is the only one that can be applied to the Win5 model. It is the most commonly used method to utilize word representations as features and our baseline. To make the comparison more fair for the Win5 model we include two additional variations that utilize both the embedding and prediction layer weights as an ensemble method for creating a word's representation:

- *Win5 AvgE*: Averaged embeddings from the embedding and prediction layer weights of Win5 skipgram:

$$\mathbf{x} = \frac{1}{2}(\mathbf{v_w} + \mathbf{v_{w'}}) \tag{13}$$

- *Win5 ConcE*: Another ensemble made by concatenating word embeddings from the embedding and prediction layer weights of the Win5 model:

$$\mathbf{x} = \mathbf{v_w} \oplus \mathbf{v_{w'}} \tag{14}$$

Ensemble techniques have been reported to outperform simple word representations in some word similarity tasks (Levy et al., 2015). Since the EXT skipgram version uses symmetric weight matrices for the embedding and prediction layer, ensemble methods like the above could also be applied, but are not considered for these experiments. Note that contrary to the dependency based models, these ensemble methods do not create context specific representations.

The dependency graph's node representations are used as a sequence of embeddings respecting the order of the sentence to become the input for the CNN and LSTM. For the SVM BoE, word and dependency contexts of the whole sentence are averaged separately for the *Words* and *Dep* method, and then averaged again for the *Wavg* method or concatenated for the *Conc* method. As we are evaluating performance of embeddings, we do not perform updates during training of CNNs and LSTMs.

5.2 Datasets and Results

TREC Question Classification The TREC Question Classification dataset (Li and Roth, 2002) consists of 5452 train questions and 500 test questions. The task is to classify each question with one of six labels (e.g. location, definition, ...) depending on the answer they seek for. For CNNs and LSTMs 10% of the training data were used as the dev set to pick the best model among different iterations. Classification accuracy results for each input representations and classification method can be seen on Table 2.

Embeddings	SVM	CNN	LSTM
Win5 Words	81.4	92.8	88.4
Win5 AvgE	81.4	91.2	88.8
Win5 ConcE	82.4	92.6	90.4
LG Words	86.8	93.8	90.6
LG Dep	85.2	89.0	87.2
LG Wavg	87.2	93.4	91.2
LG Conc	84.0	94.6	92.0
EXT Words	88.4	94.2	91.8
EXT Dep	87.6	90.6	89.8
EXT Wavg	89.0	**95.0**	92.2
EXT Conc	**91.6**	93.2	**94.4**
tree CNN		**96.0**	

Table 2: Accuracy on 6-way TREC question classification task. Tree CNN is the state-of-the-art result for this task (Mou et al., 2015).

SST-2 The Stanford Sentiment Treebank dataset (Socher et al., 2013) has fine grained sentiment polarity scores for movie reviews on phrasal and sentence level. The binary version of the task considers only positive and negative sentiment labels, resulting in a 6920/872/1821 split for training/dev/testing sets. All the models were trained using only the sentence level annotations. Classification accuracies for all models are reported on Table 3.

Embeddings	SVM	CNN	LSTM
Win5 Words	80.1	83.5	76.1
Win5 AvgE	79.5	83.2	76.9
Win5 ConcE	80.3	82.9	77.6
LG Words	78.5	84.5	77.2
LG Dep	76.0	76.8	69.1
LG Wavg	78.9	82.0	78.6
LG Conc	79.8	82.7	79.7
EXT Words	80.5	84.1	77.6
EXT Dep	77.7	77.2	69.6
EXT Wavg	**80.6**	**84.6**	75.7
EXT Conc	**80.6**	83.5	**79.8**
CNN-multichannel		**88.1**	

Table 3: Accuracy on Stanford Sentiment Treebank binary classification task. CNN-multichannel is the best result reported in Kim (2014). This result uses phrase-level annotations resulting in a much larger training dataset than the other models in the table.

SemEval 2010 Relation Identification The SemEval 2010 Relation Identification task (Hendrickx et al., 2009) considers the classification of semantic relations between pairs of nominals into 19 classes. The classes are formed by 9 types of relations (e.g. cause-effect, component-whole, ...) with directionality taken into account and an extra OTHER class. We only used the shortest dependency path between the two nominals as the input to classifiers. On table 4 we report results using the official SemEval metric of macro-averaged F1-Score for (9+1)-way classification, taking directionality into account.

6 Discussion

Our evaluation shows that dependency context embeddings can provide valuable syntactic information for sentence classification tasks using the 3 classification methods described. Out of the three tasks, Question Classification and Relation Identification showed great improvements when using dependency

Embeddings	**SVM**	**CNN**	**LSTM**
Win5 Words	72.23	81.60	77.30
Win5 AvgE	71.09	79.46	76.67
Win5 ConcE	72.74	81.33	78.09
LG Words	75.29	84.18	79.94
LG Dep	75.19	79.13	74.77
LG Wavg	77.61	83.17	79.69
LG Conc	**78.71**	83.41	78.57
EXT Words	74.93	83.69	80.24
EXT Dep	75.64	79.30	75.64
EXT Wavg	77.42	**84.31**	79.59
EXT Conc	78.53	83.93	**80.53**
CNN-NS	**85.6**		

Table 4: F1 score for SemEval 2010 Relation Identification task. CNN-NS are CNNs with negative sampling that achieve state-of-the-art result for this task (Xu et al., 2015).

context embeddings compared to the baseline, while sentiment classification only showed moderate improvements. This is in agreement with previous research (Li et al., 2015), where explicit syntactic information was provided to classifiers by using tree structured networks and showed that syntax is not crucial for SST.

It is notable that for QC and RI, using only word embeddings that are trained with syntactic information (LG and EXT models) still outperforms the baseline window based skipgram. Using the dependency context embeddings as a means to represent the dependency parse of sentences consistently outperforms the baseline method across the three tasks and for every classification method. This indicates that this additional syntactic information cannot be recovered by the CNN and LSTM even though they have access to the sequential structure of sentences. As expected, the SVM BoE benefit the most by the addition of dependency context embeddings since these are its only source of structural information.

The dependency context embeddings from the EXT model outperform the LG model, both when used alone and when in combination with the word embeddings. This can be attributed to the additional information they are exposed to during training.

The effectiveness of the *Wavg* compared to the *Conc* method for combining word and dependency context embeddings seems to depend on the classification method. In genearal, we observe that the CNN performs better with Wavg, while SVM and LSTM with Conc. On the other hand, the ensemble methods of the Win5 model (*AvgE* and *ConcE*) do not provide any consistent advantage over the baseline. In most cases, *AvgE* slightly hurts performance while *ConcE* slighty improves it.

While the purpose of our experiments is a comparison of embeddings and little hyperparameter tuning was done for the classifiers, results of the CNN using EXT Wavg representations for QC (95.0) and RI (84.31) are close to the best reported results for these tasks: 96.0 for QC (Mou et al., 2015) and 85.6 for RI (Xu et al., 2015) . These results are obtained by dependency tree based CNNs for QC, and CNNs with additional training by negative sampling and wordnet features for RI. It will be interesting to see if such architectures can further improve using dependency based representations.

All our trained embeddings and evaluation code will be made publicly available.

7 Conclusions

We compare a window based, a dependency based and an extended dependency based skipgram model in word similarity and sentence classification tasks of question classification, binary sentiment prediction and semantic relation identification. For sentence classification, we use three classifiers (SVM, CNN, LSTM) and experiment with several methods of utilizing dependency context feature embeddings to create representations that capture the syntactic role of words in dependency graphs. We show that dependency based models produce word embeddings that better capture functional properties of words and that window based models better capture topical similarity. The dependency based word embeddings improved the performance of the three classifiers for question classification and semantic relation identification, but not for sentiment prediction. Finally, using dependency context features along with the word embeddings we observed better performance for the three classifiers in each task.

References

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Marco Baroni, The Pham Nghia, Germán Kruszewski, and Angeliki Lazaridou. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model.

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)*, volume 1, pages 740–750.

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. *arXiv preprint arXiv:1508.02354*.

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.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC*, pages 4585–4592.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly learning word representations and composition functions using predicate-argument structures. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1544–1555.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *arXiv preprint arXiv:1408.3456*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

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, pages 302–308.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Jiwei Li, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL), Denver, CO*.

Oren Melamud, Omer Levy, Ido Dagan, and Israel Ramat-Gan. 2015. A simple word embedding model for lexical substitution.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of ICLR Workshop*.

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

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *In Proceedings of the International Conference on Machine Learning*.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *Unpublished manuscript: http://arxiv. org/abs/1504. 01106v5. Version*, 5.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical*

Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211. Association for Computational Linguistics.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

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*, pages 384–394. Association for Computational Linguistics.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. *arXiv preprint arXiv:1506.07650*.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.