Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology, San Diego, CA, June 2016.

The Role of Context Types and Dimensionality in Learning Word Embeddings

Oren Melamud[†*] **David McClosky**[‡*] **Siddharth Patwardhan**[◇] **Mohit Bansal**[§]

[†]Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel
melamuo@cs.biu.ac.il

[‡]Google, New York, NY, USA
dmcc@google.com

[◇]IBM Watson, Yorktown Heights, NY, USA
siddharth@us.ibm.com

[§]Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
mbansal@ttic.edu

Abstract

We provide the first extensive evaluation of how using different types of context to learn skip-gram word embeddings affects performance on a wide range of intrinsic and extrinsic NLP tasks. Our results suggest that while intrinsic tasks tend to exhibit a clear preference to particular types of contexts and higher dimensionality, more careful tuning is required for finding the optimal settings for most of the extrinsic tasks that we considered. Furthermore, for these extrinsic tasks, we find that once the benefit from increasing the embedding dimensionality is mostly exhausted, simple concatenation of word embeddings, learned with different context types, can yield further performance gains. As an additional contribution, we propose a new variant of the skip-gram model that learns word embeddings from weighted contexts of substitute words.

1 Introduction

Word embeddings have become increasingly popular lately, proving to be valuable as a source of features in a broad range of NLP tasks with limited supervision (Turian et al., 2010; Collobert et al., 2011; Socher et al., 2013; Bansal et al., 2014). `word2vec`[1] skip-gram (Mikolov et al., 2013a) and `GloVe`[2] (Pennington et al., 2014) are among the most widely used word embedding models today. Their success is largely due to an efficient and user-friendly implementation that learns high-quality word embeddings from very large corpora.

Both `word2vec` and `GloVe` learn low-dimensional continuous vector representations for words by considering window-based contexts, i.e., context words within some fixed distance of each side of the target word. However, the underlying models are equally applicable to different choices of context types. For example, Bansal et al. (2014) and Levy and Goldberg (2014) showed that using syntactic contexts rather than window contexts in `word2vec` captures *functional* similarity (as in *lion:cat*) rather than *topical* similarity or *relatedness* (as in *lion:zoo*). Further, Bansal et al. (2014) and Melamud et al. (2015b) showed the benefits of such modified-context embeddings in dependency parsing and lexical substitution tasks. However, to the best of our knowledge, there has not been an extensive evaluation of the effect of multiple, diverse context types on a wide range of NLP tasks.

Word embeddings are typically evaluated on *intrinsic* and *extrinsic* tasks. Intrinsic tasks mostly include predicting human judgments of semantic relations between words, e.g., as in WordSim-353 (Finkelstein et al., 2001), while extrinsic tasks include various 'real' downstream NLP tasks, such as coreference resolution and sentiment analysis. Re-

[*]Majority of work performed while at IBM Watson.

[1]`http://code.google.com/p/word2vec/`

[2]`http://nlp.stanford.edu/projects/glove/`

cent works have shown that while intrinsic evaluations are easier to perform, their correlation with results on extrinsic evaluations is not very reliable (Schnabel et al., 2015; Tsvetkov et al., 2015), stressing the importance of the latter.

In this work, we provide the first extensive evaluation of word embeddings learned with different types of context, on a wide range of intrinsic similarity and relatedness tasks, and extrinsic NLP tasks, namely dependency parsing, named entity recognition, coreference resolution, and sentiment analysis. We employ contexts based of different word window sizes, syntactic dependencies, and a lesser-known substitute words approach (Yatbaz et al., 2012). Finally, we experiment with combinations of the above word embeddings, comparing two approaches: (1) simple vector concatenation that offers a wider variety of features for a classifier to choose and learn weighted combinations from, and (2) dimensionality reduction via either Singular Value Decomposition or Canonical Correlation Analysis, which tries to find a smaller subset of features.

Our results suggest that it is worthwhile to carefully choose the right type of word embeddings for an extrinsic NLP task, rather than rely on intrinsic benchmark results. Specifically, picking the optimal context type and dimensionality is critical. Furthermore, once the benefit from increasing the embedding dimensionality is mostly exhausted, concatenation of word embeddings learned with different context types can yield further performance gains.

2 Word Embedding Context Types

2.1 Learning Corpus

We use a fixed learning corpus for a fair comparison of all embedding types: a concatenation of three large English corpora: (1) English Wikipedia 2015, (2) UMBC web corpus (Han et al., 2013), and (3) English Gigaword (LDC2011T07) newswire corpus (Parker et al., 2011). Our concatenated corpus is diverse and substantial in size with approximately 10B words. This allows us to learn high quality embeddings that cover a large vocabulary. After extracting clean text from these corpora, we used Stanford CoreNLP (Manning et al., 2014) for sentence splitting, tokenization, part-of-speech tagging and dependency parsing.[3] Then, all tokens were lower-cased, and sentences were shuffled to prevent structured bias. When learning word embeddings, we ignored words with corpus frequency lower than 100, yielding a vocabulary of about 500K words.[4]

2.2 Window-based Word Embeddings

We used `word2vec`'s skip-gram model with negative sampling (Mikolov et al., 2013b) to learn window-based word embeddings.[5] This popular method embeds both target words and contexts in the same low-dimensional space, where the embeddings of a target and context are pushed closer together the more frequently they co-occur in a learning corpus. Indirectly, this also results in similar embeddings for target words that co-occur with similar contexts. More formally, this method optimizes the following objective function:

$$L = \sum_{(t,c) \in PAIRS} L_{t,c} \tag{1}$$

$$L_{t,c} = \log \sigma(v'_c \cdot v_t) + \sum_{neg \in NEGS_{(t,c)}} \log \sigma(-v'_{neg} \cdot v_t) \tag{2}$$

where v_t and v'_c are the vector representations of target word t and context word c. $PAIRS$ is the set of window-based co-occurring target-context pairs considered by the model that depends on the window size, and $NEGS_{(t,c)}$ is a set of randomly sampled context words used with the pair (t, c).[6]

We experimented with window sizes of 1, 5, and 10, and various dimensionalities. We denote a window-based word embedding with window size of n and dimensionality of m with Wn^m. For example, W5^{300} is a word embedding learned using a window size of 5 and dimensionality of 300.

2.3 Dependency-based Word Embeddings

We used `word2vecf`[7] (Levy and Goldberg, 2014), to learn dependency-based word embeddings from

[3]Parses follow the Universal Dependencies formalism and were produced by Stanford CoreNLP, version 3.5.2

[4]Our word embeddings are available at: `www.cs.biu.ac.il/nlp/resources/downloads/embeddings-contexts/`

[5]We used negative sampling = 5 and iterations = 3 in all of the experiments described in this paper.

[6]For more details refer to Mikolov et al. (2013b).

[7]`http://bitbucket.org/yoavgo/word2vecf`

the parsed version of our corpus, similar to the approach of Bansal et al. (2014). `word2vecf` accepts as its input arbitrary target-context pairs. In the case of dependency-based word embeddings, the context elements are the syntactic contexts of the target word, rather than the words in a window around it. Specifically, following Levy and Goldberg (2014), we first 'collapsed' prepositions (as implemented in `word2vecf`). Then, for a target word t with modifiers m_1,...,m_k and head h, we paired the target word with the context elements (m_1, r_1),...,(m_k, r_k),(h, r_h^{-1}), where r is the type of the dependency relation between the head and the modifier (e.g., *dobj*, *prep_of*) and r^{-1} denotes an inverse relation. We denote a dependency-based word embedding with dimensionality of m by DEP^m. We note that under this setting `word2vecf` optimizes the same objective function described in Equation (1), with $PAIRS$ now comprising dependency-based pairs instead of window-based ones.

2.4 Substitute-based Word Embeddings

Substitute vectors are a recent approach to representing contexts of target words, proposed in Yatbaz et al. (2012). Instead of the neighboring words themselves, a substitute vector includes the potential filler words for the target word slot, weighted according to how 'fit' they are to fill the target slot *given* the neighboring words. For example, the substitute vector representing the context of the word *love* in "I love my job", could look like: [*quit* 0.5, *love* 0.3, *hate* 0.1, *lost* 0.1]. Substitute-based contexts are generated using a language model and were successfully used in distributional semantics models for part-of-speech induction (Yatbaz et al., 2012), word sense induction (Baskaya et al., 2013), functional semantic similarity (Melamud et al., 2014) and lexical substitution tasks (Melamud et al., 2015a).

Similar to Yatbaz et al. (2012), we consider the words in a substitute vector, as a weighted set of contexts 'co-occurring' with the observed target word. For example, the above substitute vector is considered as the following set of weighted target-context pairs: {(*love*, *quit*, 0.5), (*love*, *love*, 0.3), (*love*, *hate*, 0.1), (*love*, *lost*, 0.1)}. To learn word embeddings from such weighted target-context pairs, we extended `word2vecf` by modifying the objective function in Equation (1) as follows:

$$L = \sum_{(t,c) \in PAIRS} \alpha_{t,c} \cdot L_{t,c} \quad (3)$$

where $\alpha_{t,c}$ is the weight of the target-context pair (t, c). With this simple modification, the effect of target-context pairs on the learned word representations becomes proportional to their weights.

To generate the substitute vectors we followed the methodology in (Yatbaz et al., 2012; Melamud et al., 2015a). We learned a 4-gram Kneser-Ney language model from our learning corpus using KenLM (Heafield et al., 2013). Then, we used FASTSUBS (Yuret, 2012) with this language model to efficiently generate substitute vectors, where the weight of each substitute s is the conditional probability $p(s|C)$ for this substitute to fill the target slot given the sentential context C. For efficiency, we pruned the substitute vectors to their top-10 substitutes, $s_1..s_{10}$, and normalized their probabilities such that $\sum_{i=1..10} p(s_i|C) = 1$. We also generated only up to 20,000 substitute vectors for each target word type. Finally, we converted each substitute vector into weighted target-substitute pairs and used our extended version of `word2vecf` to learn the substitute-based word embeddings, denoted SUB^m.

W10^{300}	DEP^{300}	SUB^{300}
played	play	singing
play	played	rehearsing
plays	understudying	performing
professionally	caddying	composing
player	plays	running

Table 1: The top five words closest to target word *playing* in different embedding spaces.

2.5 Qualitative Effect of Context Type

To motivate the rest of our work, we first qualitatively inspect the top most-similar words to some target words, using cosine similarity of their respective embeddings. As illustrated in Table 1, in embeddings learned with large window contexts, we see both functionally similar words and topically similar words, sometimes with a different part-of-speech. With small windows and dependency contexts, we generally see much fewer topically similar words, which is consistent with previous findings (Bansal et al., 2014; Levy and Goldberg, 2014). Finally, with substitute-based contexts, there appears to be even a stronger preference for functional similarity, with a tendency to also strictly preserve verb tense.

3 Word Embedding Combinations

As different choices of context type yield word embeddings with different properties, we hypothesize that combinations of such embeddings could be more informative for some extrinsic tasks. We experimented with two alternative approaches to combine different sets of word embeddings: (1) Simple vector concatenation, which is a lossless combination that comes at the cost of increased dimensionality, and (2) SVD and CCA, which are lossy combinations that attempt to capture the most useful information from the different embeddings sets with lower dimensionality. The methods used are described in more detail next.

3.1 Concatenation

Perhaps the simplest way to combine two different sets of word embeddings (sharing the same vocabulary) is to concatenate their word vectors for every word type. We denote such a combination of word embedding set A with word embedding set B using the symbol (+). For example W10+DEP^{600} is the concatenation of W10^{300} with DEP^{300}. Naturally, the dimensionality of the concatenated embeddings is the sum of the dimensionalities of the component embeddings. In our experiments, we only ever combine word embeddings of equal dimensionality.

The motivation behind concatenation relates primarily to supervised models in extrinsic tasks. In such settings, we hypothesize that using concatenated word embeddings as input features to a classifier could let it choose and combine (i.e., via learned weights) the most suitable features for the task. Consider a situation where the concatenated embedding W10+DEP^{600} is used to represent the word inputs to a named entity recognition classifier. In this case, the classifier could choose, for instance, to represent entity words mostly with dependency-based embedding features (reflecting functional semantics), and surrounding words with large window-based embedding features (reflecting topical semantics).

3.2 Singular Value Decomposition

Singular Value Decomposition (SVD) has been shown to be effective in compressing sparse word representations (Levy et al., 2015). In this work, we use this technique in the same way to reduce the dimensionality of concatenated word embeddings.

3.3 Canonical Correlation Analysis

Recent work used Canonical Correlation Analysis (CCA) to derive an improved set of word embeddings. The main idea is that two distinct sets of word embeddings, learned with different types of input data, are considered as multi-views of the same vocabulary. Then, CCA is used to project each onto a lower dimensional space, where correlation between the two is maximized. The correlated information is presumably more reliable. Dhillon et al. (2011) considered their two CCA views as embeddings learned from the left and from the right context of the target words, showing improvements on chunking and named entity recognition. Faruqui and Dyer (2014) and Lu et al. (2015) considered multilingual views, showing improvements in several intrinsic tasks, such as word and phrase similarity.

Inspired by this prior work, we consider pairs of word embedding sets, learned with different types of context, as different views and correlate them using linear CCA.[8] We use either the SimLex-999 or WordSim-353-R intrinsic benchmark (section 4.1) to tune the CCA hyperparameters[9] with the Spearmint Bayesian optimization tool[10] (Snoek et al., 2012). This results in different projections for each of these tuning objectives, where SimLex-999/WordSim-353-R is expected to give some bias towards functional/topical similarity, respectively.

4 Evaluation

4.1 Intrinsic Benchmarks

We employ several commonly used intrinsic benchmarks for assessing how well word embeddings mimic human judgements of semantic similarity of words. The popular **WordSim-353** dataset (Finkelstein et al., 2001) includes 353 word pairs manually

[8]See Faruqui and Dyer (2014), Lu et al. (2015) for details.

[9]These are projection dimensionality and regularization.

[10]`github.com/JasperSnoek/spearmint`

annotated with a degree of similarity. For example, *computer*:*keyboard* is annotated with 7.62, indicating a relatively high degree of similarity. While WordSim-353 does not make a distinction between different 'flavors' of similarity, Agirre et al. (2009) proposed two subsets of this dataset, **WordSim-353-S** and **WordSim-353-R**, which focus on functional and topical similarities, respectively. **SimLex-999** (Hill et al., 2014) is a larger word pair similarity dataset with 999 annotated pairs, purposely built to focus on functional similarity. We evaluate our embeddings on these datasets by computing a score for each pair as the cosine similarity of two word vectors. The Spearman's correlation[11] between the ranking of word pairs induced from the human annotations and that from the embeddings is reported.

The **TOEFL** task contains 80 synonym selection items, where a synonym of a target word is to be selected out of four possible choices. We report the overall accuracy of a system that uses cosine distance between the embeddings of the target word and each of the choices to select the one most similar to the target word as the answer.

4.2 Extrinsic Benchmarks

The following four diverse downstream NLP tasks serve as our extrinsic benchmarks.[12]

1) Dependency Parsing (PARSE) The Stanford Neural Network Dependency (NNDEP) parser (Chen and Manning, 2014) uses dense continuous representations of words, parts-of-speech and dependency labels. While it can learn these representations entirely during the training on labeled data, Chen and Manning (2014) show that initialization with word embeddings, which were pre-trained on unlabeled data, yields improved performance. Hence, we used our different types of embeddings to initialize the NNDEP parser and compared their performance on a standard Penn Treebank benchmark. We used WSJ sections 2–21 for training and 22 for development. We used predicted tags produced via 20-fold jackknifing on sections 2–21 with the Stanford CoreNLP tagger.

[11]We used `spearmanr`, SciPy version 0.15.1.

[12]Since our goal is to explore performance trends, we mostly experimented with the tasks' development sets.

2) Named Entity Recognition (NER) We used the NER system of Turian et al. (2010), which allows adding word embedding features (on top of various other features) to a regularized averaged perceptron classifier, and achieves near state-of-the-art results using several off-the-shelf word representations. We varied the type of word embeddings used as features when training the NER model, to evaluate their effect on NER benchmarks results. Following Turian et al. (2010), we used the CoNLL-2003 shared task dataset (Tjong Kim Sang and De Meulder, 2003) with 204K/51K train/dev words, as our main benchmark. We also performed an out-of-domain evaluation, using CoNLL-2003 as the train set and the MUC7 formal run (59K words) as the test set.[13]

3) Coreference Resolution (COREF) We used the Berkeley Coreference System (Durrett and Klein, 2013), which achieves near state-of-the-art results with a log-linear supervised model. Most of the features in this model are associated with pairs of *current* and *antecedent* reference mentions, for which a coreference decision needs to be made. To evaluate the contribution of different word embedding types to this model, we extended it to support the following additional features: $\{a_i\}_{i=1..m}$, $\{c_i\}_{i=1..m}$ and $\{a_i \cdot c_i\}_{i=1..m}$, where a_i or c_i is the value of the ith dimension in a word embedding vector representing the antecedent or current mention, respectively. We considered two different word embedding representations for a mention: (1) the embedding of the head word of the mention and (2) the average embedding of all words in the mention. The features of both types of representations were presented to the learning model as inputs at the same time. They were added on top of Berkeley's full feature list ('FINAL') as described in Durrett and Klein (2013). We evaluated our features on the CoNLL-2012 coreference shared task (Pradhan et al., 2012).

4) Sentiment Analysis (SENTI) Following Faruqui et al. (2014), we used a sentence-level binary decision version of the sentiment analysis task from Socher et al. (2013). In this setting, neutral sentences were discarded and all remaining sentences were labeled coarsely as positive or negative. Maintaining the original split into train/dev

[13]See Turian et al. (2010) for more details on this setting.

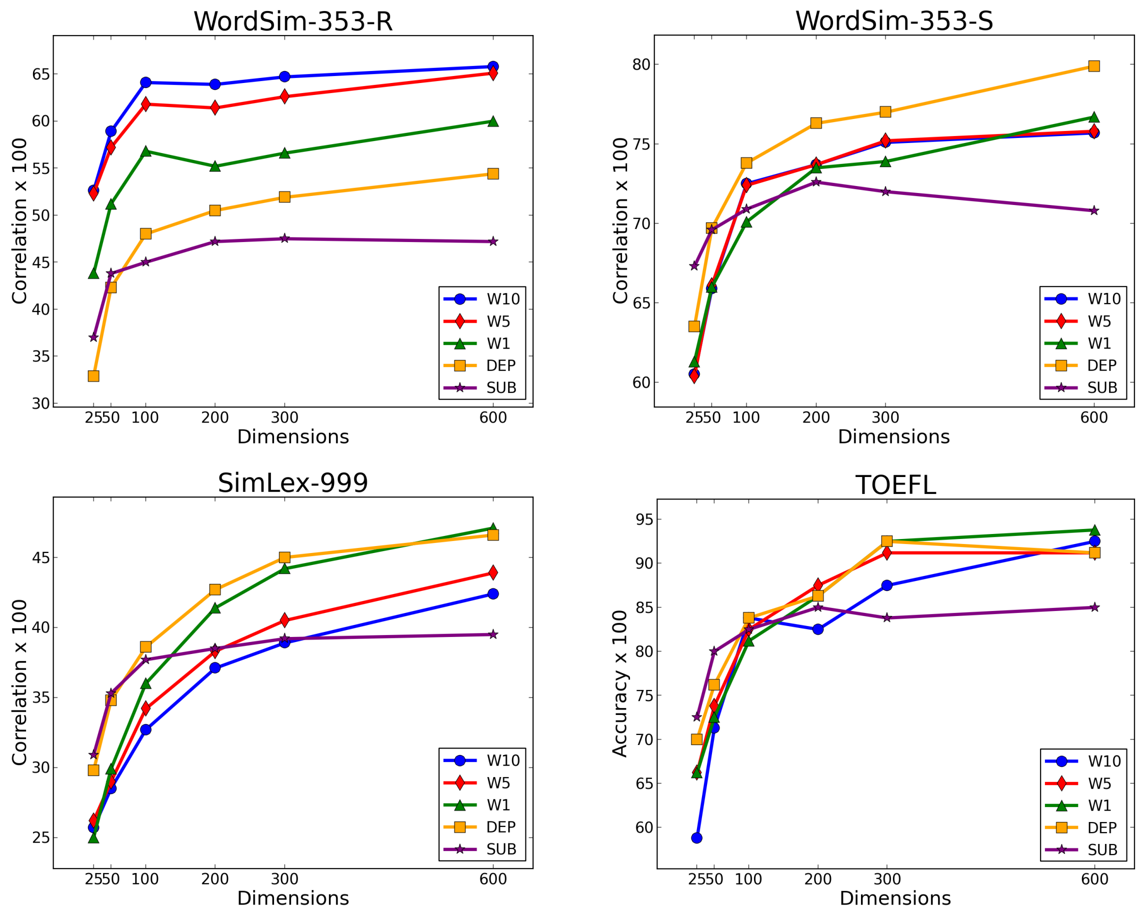


Figure 1: Intrinsic tasks' results for embeddings learned with different types of contexts.

results, we get a dataset containing 6920/872 sentences. To evaluate different types of word embeddings, we represented each sentence as an average of its word embeddings and then used an L2-regularized logistic regression classifier trained on these features to predict the sentiment labels.

5 Results

5.1 Intrinsic Results for Context Types

The results on the intrinsic tasks are illustrated in Figure 1. First, we see that the performance on all tasks generally increases with the number of dimensions, reaching near-optimal performance at around 300 dimensions, for all types of contexts. This is in line with similar observations on skip-gram word embeddings (Mikolov et al., 2013a).

Looking further, we observe that there are significant differences in the results when using different types of contexts. The effect of context choice is perhaps most evident in the WordSim-353-R task, which captures topical similarity. As might be expected, in this benchmark, the largest-window word embeddings perform best. The performance decreases with the decrease in window size and then reaches significantly lower levels for dependency (DEP) and substitute-based (SUB) embeddings. Conversely, in WordSim-353-S and SimLex-999, both of which capture a more functional similarity, the DEP embeddings are the ones that perform best, strengthening similar observations in Levy and Goldberg (2014). Finally, in the TOEFL benchmark, all contexts except for SUB, perform comparably.

5.2 Extrinsic Results for Context Types

The extrinsic tasks results are illustrated in Figure 2. A first observation is that optimal extrinsic results may be reached with as few as 50 dimensions. Furthermore, performance may even degrade when us-

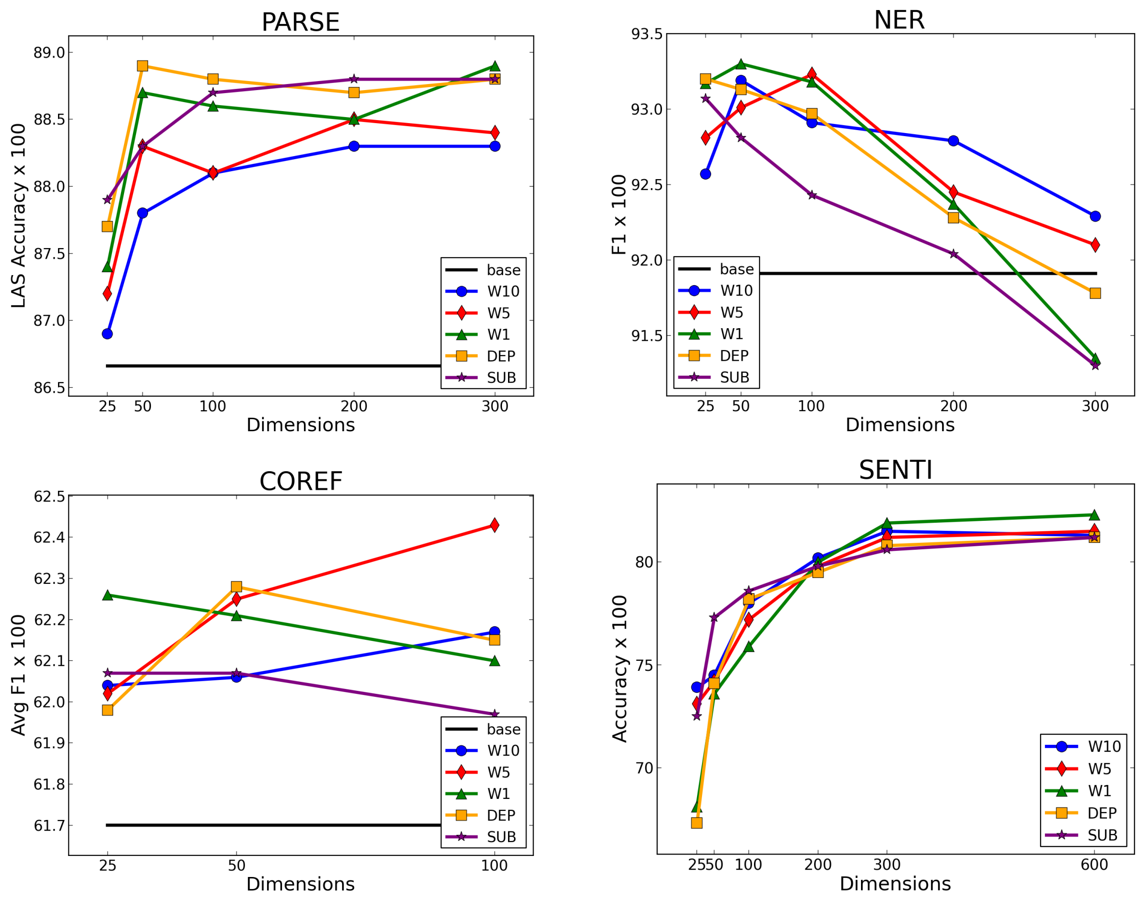


Figure 2: Extrinsic tasks' development set results for embeddings learned with different types of contexts. 'base' denotes the results with no word embedding features. Due to computational limitations we tested NER and PARSE with only up to 300 dimensions embeddings, and COREF with up to 100.

ing too many dimensions, as is most evident in the NER task. This behavior presumably depends on various factors, such as the size of the labeled training data or the type of classifier used, and highlights the importance of tuning the dimensionality of word embeddings in extrinsic tasks. This is in contrast to intrinsic tasks, where higher dimensionality typically yields better results.

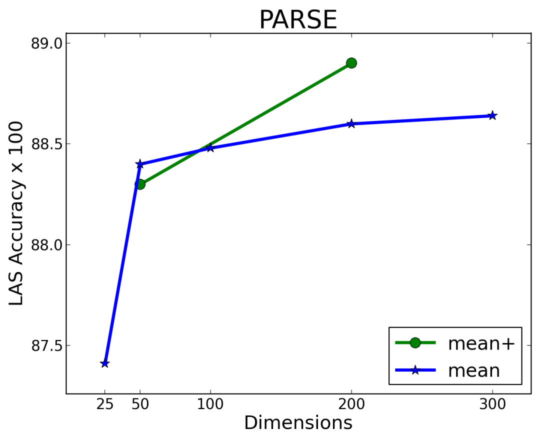
Next, comparing the results of different types of contexts, we see, as might be expected, that dependency embeddings work best in the PARSE task. More generally, embeddings that do well in functional similarity intrinsic benchmarks and badly in topical ones (DEP, SUB and W1) work best for PARSE, while large window contexts perform worst, similar to observations in Bansal et al. (2014).

In the rest of the tasks it's difficult to say which context works best for what. One possible explanation to this in the case of NER and COREF is that the embedding features are used as add-ons to an already competitive learning system. Therefore, the total improvement on top of a 'no embedding' baseline is relatively small, leaving little room for significant differences between different contexts.

Context type	F1 x 100
DEP	79.8
W1	79.3
SUB	79.0
W10	78.1
W5	77.4
None	71.8

Table 2: NER MUC out-of-domain results for different embeddings with dimensionality = 25.

We did find a more notable contribution of word



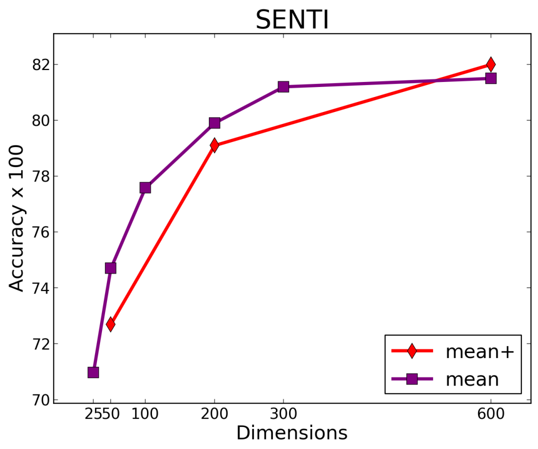


Figure 3: Mean development set results for the tasks PARSE and SENTI. ‘mean’ and ’mean+’ stand for mean results across all single context types and context concatenations, respectively.

embedding features to the overall system performance in the out-of-domain NER MUC evaluation, described in Table 2. In this out-of-domain setting, all types of contexts achieve at least five points improvement over the baseline. Presumably, this is because continuous word embedding features are more robust to differences between train and test data, such as the typical vocabulary used. However, a detailed investigation of out-of-domain settings is out of scope for this paper and left for future work.

5.3 Extrinsic Results for Combinations

A comparison of the results obtained on the extrinsic tasks using the word embedding concatenations (*concats*), described in section 3.1, versus the original single context word embeddings (*singles*), appears in Table 3. To control for dimensionality, concats are always compared against singles with identical dimensionality. For example, the 200-dimensional concat W10+DEP^{200}, which is a concatenation of W10^{100} and DEP^{100}, is compared against 200-dimensional singles, such as W10^{200}.

Looking at the results, it seems like the benefit from concatenation depends on the dimensionality and task at hand, as also illustrated in Figure 3. Given task X and dimensionality d, if $\frac{d}{2}$ is in the range where increasing the dimensionality yields significant improvement on task X, then it’s better to simply increase dimensionality of singles from $\frac{d}{2}$ to d rather than concatenate. The most evident example for this are the results on the SENTI task with $d = 50$. In this case, the benefit from concatenating two 25-dimensional singles is notably lower than that of using a single 50-dimensional word embedding. On the other hand, if $\frac{d}{2}$ is in the range where near-optimal performance is reached on task X, then concatenation seems to pay off. This can be seen in SENTI with $d = 600$, PARSE with $d = 200$, and NER with $d = 50$. More concretely, looking at the best performing concatenations, it seems like combinations of the topical W10 embedding with one of the more functional ones, SUB, DEP or W1, typically perform best, suggesting that there is added value in combining embeddings of different nature.

Finally, our experiments with the methods using SVD (section 3.2) and CCA (section 3.3) yielded degraded performance compared to single word embeddings for all extrinsic tasks and therefore are not reported for brevity. These results seem to further strengthen the hypothesis that the information captured with varied types of context is different and complementary, and therefore it is beneficial to preserve these differences as in our concatenation approach.

6 Related Work

There are a number of recent works whose goal is a broad evaluation of the performance of different word embeddings on a range of tasks. However, to the best of our knowledge, none of them focus on embeddings learned with diverse context types as we do. Levy et al. (2015), Lapesa and Evert (2014), and Lai et al. (2015) evaluate several design choices when learning word representations. However, Levy et al. (2015) and Lapesa and Evert (2014)

Dimensions	Result	SENTI	PARSE	NER	COREF
50	best+	74.3 (W10+W1)	88.7 (W10+SUB)	**93.6** (W1+DEP)	**62.4** (W10+W1)
	best	**77.3** (SUB)	**88.9** (W1)	93.3 (W1)	62.3 (DEP)
	mean+	72.7	88.3	**93.3**	62.1
	mean	**74.7**	**88.4**	93.1	**62.2**
200	best+	**81.0** (W10+SUB)	**89.1** (W1+DEP)	**93.1** (W10+DEP)	
	best	80.2 (W10)	88.8 (SUB)	92.8 (W10)	
	mean+	79.1	**88.9**	**92.8**	
	mean	**79.9**	88.6	92.4	
600	best+	**82.6** (W10+SUB)			
	best	82.3 (W1)			
	mean+	**82.0**			
	mean	81.5			

Table 3: Extrinsic tasks development set results obtained with word embeddings concatenations. 'best' and 'best+' are the best results achieved across all single context types and context concatenations, respectively (best performing embedding indicated in parenthesis). 'mean' and 'mean+' are the mean results for the same. Due to computational limitations of the employed systems, some of the evaluations were not performed.

perform only intrinsic evaluations and restrict context representation to word windows, while Lai et al. (2015) do perform extrinsic evaluations, but restrict their context representation to a word window with the default size of 5. Schnabel et al. (2015) and Tsvetkov et al. (2015) report low correlation between intrinsic and extrinsic results with different word embeddings (they did not evaluate different context types), which is consistent with differences we found between intrinsic and extrinsic performance patterns in all tasks, except parsing. Bansal et al. (2014) show that functional (dependency-based and small-window) embeddings yield higher parsing improvements than topical (large-window) embeddings, which is consistent with our findings.

Several works focus on particular types of contexts for learning word embeddings. Cirik and Yuret (2014) investigates S-CODE word embeddings based on substitute word contexts. Ling et al. (2015b) and Ling et al. (2015a) propose extensions to the standard window-based context modeling. Alternatively, another recent popular line of work (Faruqui et al., 2014; Kiela et al., 2015) attempts to improve word embeddings by using manually-constructed resources, such as WordNet. These techniques could be complementary to our work. Finally, Yin and Schütze (2015) and Goikoetxea et al. (2016) propose word embeddings combinations, using methods such as concatenation and CCA, but evaluate mostly on intrinsic tasks and do not consider different types of contexts.

7 Conclusions

In this paper we evaluated skip-gram word embeddings on multiple intrinsic and extrinsic NLP tasks, varying dimensionality and type of context. We show that while the best practices for setting skip-gram hyperparameters typically yield good results on intrinsic tasks, success on extrinsic tasks requires more careful thought. Specifically, we suggest that picking the optimal dimensionality and context type are critical for obtaining the best accuracy on extrinsic tasks and are typically task-specific. Further improvements can often be achieved by combining complementary word embeddings of different context types with the right dimensionality.

Acknowledgments

We thank Do Kook Choe for providing us the jackknifed version of WSJ. We also wish to thank the IBM Watson team for helpful discussions and our anonymous reviewers for their comments. This work was partially supported by the Israel Science Foundation grant 880/12 and the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

[Agirre et al.2009] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of NAACL*. Association for Computational Linguistics.

[Bansal et al.2014] 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*.

[Baskaya et al.2013] Osman Baskaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. Ai-ku: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In *Proceedings of the SemEval*.

[Chen and Manning2014] 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.

[Cirik and Yuret2014] Volkan Cirik and Deniz Yuret. 2014. Substitute based scode word embeddings in supervised nlp tasks. *arXiv preprint arXiv:1407.6853*.

[Collobert et al.2011] 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.

[Dhillon et al.2011] 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.

[Durrett and Klein2013] Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proc. of EMNLP*.

[Faruqui and Dyer2014] Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*.

[Faruqui et al.2014] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. In *Proceedings of Deep Learning and Representation Learning Workshop, NIPS*.

[Finkelstein et al.2001] 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.

[Goikoetxea et al.2016] Josu Goikoetxea, Eneko Agirre, and Aitor Soroa. 2016. Single or multiple? combining word representations independently learned from text and wordnet. In *Proceedings of AAAI*.

[Han et al.2013] Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Johnathan Weese. 2013. UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, June.

[Heafield et al.2013] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of ACL*.

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

[Kiela et al.2015] Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness.

[Lai et al.2015] Siwei Lai, Kang Liu, Liheng Xu, and Jun Zhao. 2015. How to generate a good word embedding? *arXiv preprint arXiv:1507.05523*.

[Lapesa and Evert2014] Gabriella Lapesa and Stefan Evert. 2014. A large scale evaluation of distributional semantic models: Parameters, interactions and model selection. *Transactions of the Association for Computational Linguistics*, 2:531–545.

[Levy and Goldberg2014] Omer Levy and Yoav Goldberg. 2014. Dependencybased word embeddings. In *Proceedings of ACL*.

[Levy et al.2015] 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.

[Ling et al.2015a] Wang Ling, Lin Chu-Cheng, Yulia Tsvetkov, Silvio Amir, Ramón Fernandez Astudillo, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of EMNLP*.

[Ling et al.2015b] Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015b. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of NAACL-HLT*.

[Lu et al.2015] Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, , and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of NAACL*.

[Manning et al.2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J.

Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

[Melamud et al.2014] Oren Melamud, Ido Dagan, Jacob Goldberger, Idan Szpektor, and Deniz Yuret. 2014. Probabilistic modeling of joint-context in distributional similarity. In *Proceedings of CoNLL*.

[Melamud et al.2015a] Oren Melamud, Ido Dagan, and Jacob Goldberger. 2015a. Modeling word meaning in context with substitute vectors. In *Proceedings of NAACL*.

[Melamud et al.2015b] Oren Melamud, Omer Levy, and Ido Dagan. 2015b. A simple word embedding model for lexical substitution. In *Proceedings of the Vector Space Modeling for NLP Workshop, NAACL*.

[Mikolov et al.2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*.

[Mikolov et al.2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.

[Parker et al.2011] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth edition. *Linguistic Data Consortium, LDC2011T07*, June.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, volume 12.

[Pradhan et al.2012] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

[Schnabel et al.2015] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proc. of EMNLP*.

[Snoek et al.2012] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

[Socher et al.2013] 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.

[Tjong Kim Sang and De Meulder2003] Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

[Tsvetkov et al.2015] Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proc. of EMNLP*.

[Turian et al.2010] J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: A simple and general method for semisupervised learning. In *Proc. of ACL*, pages 384–394.

[Yatbaz et al.2012] Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. 2012. Learning syntactic categories using paradigmatic representations of word context. In *Proceedings of EMNLP*.

[Yin and Schütze2015] Wenpeng Yin and Hinrich Schütze. 2015. Learning word meta-embeddings by using ensembles of embedding sets. *arXiv preprint arXiv:1508.04257*.

[Yuret2012] Deniz Yuret. 2012. FASTSUBS: An efficient and exact procedure for finding the most likely lexical substitutes based on an n-gram language model. *Signal Processing Letters, IEEE*, 19(11):725–728.