Good, Better, Best: Choosing Word Embedding Context

James Cross[1] **Bing Xiang**[2] **Bowen Zhou**[2]

[1]School of EECS
Oregon State University
Corvallis, OR
crossj@oregonstate.edu

[2]IBM Watson
T. J. Watson Research Center
Yorktown Heights, NY
{bingxia,zhou}@us.ibm.com

Abstract

We propose two methods of learning vector representations of words and phrases that each combine sentence context with structural features extracted from dependency trees. Using several variations of neural network classifier, we show that these combined methods lead to improved performance when used as input features for supervised term-matching.

1 Introduction

Continuous-space embeddings of words and phrases are very appealing for many applications in NLP. When used as a feature to represent words, they are much more compact than "one-hot" vectors. In addition, their geometry can encode a wealth of information about relationships between words, and thus their meanings, which can be exploited by downstream applications.

The critical question is how to best achieve that geometry for a given task. There are two types of training data which are often used to learn word vectors. Approaches based on the "distributional hypothesis" assume that the semantics of a word can be determined through the contexts in which it appears. They model vectors through some form of dimension compression on word co-occurrence statistics taken from an example text. Other methods learn vectors from pre-existing information about specific relationships between words, such as those found in a structured knowledge base.

We propose two vector-learning methods which combine sentence-context input with structural features extracted automatically from dependency trees for the same underlying corpus. We compare them with vectors trained from only one of these types of input, and show that they lead to improved performance on simple classification tasks.

2 Related Work

Many different approaches to learning vector representations of words have been taken. Methods trained from large text examples by relying directly or indirectly on word co-occurrence counts date back to latent semantic analysis (Deerwester, 1990). Recent examples include the window-based approach of Mikolov et al. (2013c) and the GloVe method (Pennington et al., 2014). There has also been work incorporating prior knowledge with bag-of-words context (Yu and Dredze, 2014).

It is also possible to perform unsupervised learning on structured information about the underlying entities or concepts being represented, such as the relationships encoded in an online knowledge base. Approaches for utilizing this type of training data include the use of neural networks and/or kernel density estimation (Bordes et al., 2011) and encoding relations as translations on hyperplanes (Wang et al., 2014).

The most direct precursor to our work is the well-known Word2Vec algorithm (Mikolov et al., 2013a; Mikolov et al., 2013b; Mikolov et al., 2013c). It consists of a simple single-layer neural model where the goal is to maximize the ability of the word representations to predict neighbors of the word in a large raw-text corpus. It was observed by Levy and Goldberg (2014) that this algorithm could be generalized to use any type of discrete features with which words

could be paired as the "contexts" to be predicted.

3 Combined Objective

In this work, we combine sentence-window context with structural features described in Section 4. The joint objective consists of using both types of training data in parallel so that the training objective is to maximize the combined log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) + \frac{\alpha}{|P|}\sum_{(w,f)\in P}\log p(f|w) \quad (1)$$

where T is the number of words in the training corpus, c is the context window size, P is the set of all pairs of words and structural features (w, f), and α is a weighting parameter as between the two types of training data.

The conditional probabilities are modeled with vectors as in Word2Vec, and two sets of output vectors, internal to the algorithm, are learned to represent words and structural features as contexts. This objective is implemented with a joint algorithm which performs updates using the sentence context and the word/feature pairs in parallel. Both types of updates are based on the negative sampling training method (Mikolov et al., 2013c).

We also implemented a sequential algorithm where vectors trained with just sentence-window context are used as initial representations which are subsequently refined by updates using the structural features. The relative success of vectors trained in this way suggests that proceeding from a more general representation toward one which is more directly tailored for a given task is a promising avenue.

In the sections that follow, we describe a number of simple classification experiments designed to compare sets of word embeddings trained using these various methods. For the term-matching tasks we considered, the structural features alone led to better-performing vectors than those trained with just linear word context, while the combined-objective methods provided further improvement.

4 Structural Features

As a representative means of training word vectors on functional relationships between words in sentences, rather than just word proximity, we began with first-order dependency arcs of the type described in Levy and Goldberg (2013). For every dependency arc, each word is annotated with a feature which consists of the arc label, whether the word is the head or tail of the arc, and the adjoined word.[1] We also extracted two types of higher-order features from the dependency trees: noun-relationship patterns, and subject-object relationships.

Feature	Dependency Pattern	Example
such_as	$mwe(\texttt{as}_i, \texttt{such}) \wedge$ $prep(X, \texttt{as}_i) \wedge$ $pobj(\texttt{as}_i, Y)$	... **anarchists** such as Frederico **Urales** ...
known_as	$prep(\texttt{known}_i, \texttt{as}_j) \wedge$ $nsubjpass(\texttt{known}_i, X) \wedge$ $pobj(\texttt{as}_j, Y)$	The **incident** became known as the Haymarket **affair**.
name_for	$nsubj(X, \texttt{name}_i) \wedge$ $prep(\texttt{name}_i, \texttt{for}_j) \wedge$ $pobj(\texttt{for}_j, Y)$	The trivial (non-systematic) name for **alkanes** is **paraffins**.

Table 1: Examples of noun-relationship pattern features extracted from dependency structures. In each case, one feature is created for word X which includes the feature type and Y, and another is created for Y which includes the feature type and X. Subscripts indicate sentence position.

Noun-relationship features capture certain taxonomic relationships between nouns, as reflected in the structure of a sentence, such as when one word names an instance of a category defined by another word, or when they are different names for the same thing. This methodology is based on the syntactic patterns used to recognize *is-a* relationships in (Fan et al., 2011). Examples of these dependency patterns and the relationships they identify can be seen in Table 1.

In addition, second-order features directly linking subjects and objects of transitive verbs (and including the verb) are also extracted. Thus for the sentence "The woman ate the paella" the word "woman" will be annotated with a feature indicating that it is the subject of the verb "ate" *with the direct object "paella"*, and "paella" will receive a feature marking it as the object of "ate" with the subject

[1] Also included from that work and in the "Dependency" test set are features coming from the "flattening" of prepositional phrases to directly link the modified head and the object of the preposition (with the preposition itself included in the feature name).

Target Word	Raw Text (Word2Vec)	Structural Features	Joint
part-of-speech	xml-like	file-type	verb-noun
	meta-language	non-template	html-like
	sxml	single-word	multiword
	script-based	xml-style	script-based
	canonicalization	single-letter	now-deprecated
dynamic	decouples	spacial [*sic*]	nonhierarchical
	user-centric	timbral	composition-based
	gesture-based	user-centric	non-virtual
	multi-path	task-focused	ever-changing
	process-based	content-driven	human/machine
normalize	normalise	normalise	normalise
	normalizing	regularize	regularize
	normalising	exponentiate	quantify
	rebalance	recalculate	define
	normalization	quantify	homogenize

Table 2: Most similar 5 words (by cosine similarity) to several target words under different vector-training regimes.

"woman".

The rationale for using both of these types of features is that nouns contain much of the semantic nexus of a sentence, and they have direct dependency relationships with essentially all other types of words. By refining word vectors based on specific relationships between nouns, the quality of the vector representations as a whole can be improved.

The features used to train vectors for our experiments were extracted from dependency trees produced by the Stanford Neural Network Dependency Parser (Chen and Manning, 2014) using the provided pre-trained model with Stanford dependency labels. The underlying corpus for all methods was the full text of English Wikipedia as of June 15, 2014.[2]

5 Qualitative Comparison

To evaluate the relative qualities captured by the various embedding methods, we looked at the closest words to a number of different target words, just as in Levy and Goldberg (2014). Unsurprisingly, the nearest words were very similar across the different vector sets, as all were trained with the same base corpus.

In particular, jointly-trained vectors lead to similar words which are subjectively very much like those produced by the vectors trained with structural features only. There is reason to believe, however, that the joint-objective vectors capture some additional semantic nuance stemming from the intersection of (and interplay between) topical and functional similarity.

For the word "part-of-speech" the most similar words under all vector sets skew toward hyphenated or compound modifiers (though the set induced by bag-of-words training includes nominals). While all of the vector sets seem to reflect the fact that this modifier is frequently used with the word "tag(s)," the jointly-trained vectors also capture its affinity with linguistics in such similar words as "verb-noun" and "multiword." Likewise with other words such as "dynamic" and "normalize" the jointly-trained vectors show properties which are similar to those using structural features alone, but also seem to encode meaning in a somewhat more targeted way ("ever-changing"; "homogenize").

6 Experiments

In order to compare their predictive properties quantitatively, we tested word embeddings produced with each method as inputs to binary term-matching classifiers using two different data sets, described below. We used shallow neural network classifiers with three alternative architectures, with a uniform hidden-layer size of 200. The results are presented in Table 3.

For the standard multilayer perceptron ("MLP") algorithm, the vector representations of each word in a pair were concatenated to make the input to a single hidden layer with a leaky rectifier activation function (Glorot et al., 2011; Maas et al., 2013). We also used a version where the two vectors were each used as input to the same hidden layer, i.e., the weights were shared, and the hidden-layer output for each, together with their element-wise absolute-value difference, were concatenated as input to the logistic regression classifier layer ("Shared MLP").

The third classifier similarly used shared weights to process each vector input (with a hyperbolic tangent activation function), but classification was made based on the cosine similarity of the hidden-layer output, with a threshold of 0.5. In this way, the network learns how to project the vectors into a space where their angular separation is a good measure of their similarity for the task at hand. We observed that this network converged very fast during training, but had performance nearly on par with

[2] As maintained by the KOPI project (Pataki et al., 2012).

	MLP	Shared MLP	Shared Cosine
WordNet			
Raw Text (Word2Vec)	75.88	73.76	77.10
Dependency	82.71	79.56	80.74
+ Stuctural Features	82.86	80.70	81.61
Joint	82.86	80.81	81.65
Sequential	83.12	81.61	82.75
PPDB			
Raw Text (Word2Vec)	93.43	92.84	93.20
Dependency	94.88	94.05	94.06
+ Structural Features	94.82	94.00	94.09
Joint	95.52	94.63	94.51
Sequential	95.48	95.03	94.53

Table 3: Accuracy results (%) for classification of WordNet synonym pairs and PPDB lexical matches for different vector sets

more complicated models.

6.1 WordNet Synonyms

The first source of term-matching pairs we considered were synonyms from the WordNet lexical database (Miller, 1995). WordNet groups words into synsets which are groups of words (and multi-word phrases) with the same meaning. We created match data by taking all distinct pairs of words within synsets of any part of speech, and generated negative examples by random shuffling.[3]

This turns out to be a relatively difficult task with unsupervised vectors alone. A possible reason is that the vectors do not distinguish between word senses, essentially folding together all usage cases into a single representation. The WordNet data is characterized by relatively common words where the synonymy may rely on a secondary or figurative definition. Examples of this in our data are the pairs (`fastball`, `bullet`) and (`arrest`, `collar`). In particular, since the raw-text data tends to group words by domain of discourse rather than function, it makes sense that a figurative usage as for "bullet" and "collar" in those pairs might cause problems.

In this case, the structural features were much more useful than bag-of-words context. The pattern-based features also led to a clear improvement over using just the subset of first-order dependency relations, which we hypothesize is largely due to the subject-object relations. The joint vectors showed little gain over using structural features only. This seems linked to the large gap in utility between the two types of training data, and points to the importance of application domain in determining embedding methods. The sequentially trained vectors performed better across the board, however, which suggests the usefulness of "priming" in this type of vector learning algorithm.

6.2 Paraphrase Database Lexical Pairs

As another source of term-matching data, we used pairs from the small (highest-confidence) lexical paraphrase dataset of the Paraphrase Database (Ganitkevitch et al., 2013). They consist of instances where two distinct words were used to mean the same thing in bi-texts. In contrast to the WordNet pairs, this dataset skews toward less common words. It includes many instances of things like alternate spellings and alternate names (or transliterations) for proper nouns.[4]

Here, the performance difference between the raw-text and structural-feature vectors was much smaller. The combined-objective methods were better than either. This suggests that when the two types of training data have comparable degrees of utility, but work through different avenues, a combined training method can create embeddings which capture the advantages of both approaches.

7 Conclusion and Future Work

It is clear that the best method of training continuous word embeddings depends on the domain of application. When comparably useful information is encoded in both sentence context and structured representations, combining both as training data can lead to superior vector representations.

For future work, we plan to try using richer feature sets, such as knowledge-base relations, as training inputs, and explore the mathematics behind the sequential training mechanism.

[3]The WordNet data, as filtered for coverage by these vector sets, consists of 21,089 training pairs, 2,636 development pairs, and 2,637 test pairs, split evenly between matches and non-matches.

[4]The PPDB data consists of 79,696 training pairs, 9,962 validation pairs, and 9,962 test pairs, split 20%/80% between matches and non-matches.

References

[Bergstra et al.2010] James Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio. 2010. Theano: A CPU and GPU Math Expression Compiler. *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

[Bordes et al.2011] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Structured Embeddings of Knowledge Bases. *AAAI*.

[Chen and Manning2014] Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[Deerwester1990] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41.

[Fan et al.2011] James Fan, Aditya Kalyanpur, J. William Murdock, and Branimir K. Boguraev. 2011. Mining Knowledge from Large Corpora for Type Coercion in Question Answering. *Web Scale Knowledge Extraction (WEKEX) Workshop at International Semantic Web Conference.*

[Fan et al.2012] James Fan, Aditya Kalyanpur, D. C. Gondek, and David A. Ferrucci. 2012. Automatic knowledge extraction from documents. *IBM Journal of Research and Development* 56, no. 3.4: 5-1.

[Ganitkevitch et al.2013] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*

[Glorot et al.2011] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume* Vol. 15.

[Levy and Goldberg2014] Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Association for Computational Linguistics (ACL)*.

[Maas et al.2013] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Workshop on Deep Learning for Audio, Speech, and Language Processing.*.

[McCord et al.2012] Michael C. McCord, J. William Murdock, and Branimir K. Boguraev. 2012. Deep parsing in Watson. *IBM Journal of Research and Development* 56, no. 3.4: 3-1.

[Mikolov et al.2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[Mikolov et al.2013b] Tomas Mikolov, lya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Continuous Space Word Representations. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, pages 3111-3119.

[Mikolov et al.2013c] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Distributed representations of words and phrases and their compositionality. In *North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*

[Miller1995] George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38.11: 39-41.

[Pataki et al.2012] Máté Pataki, Miklós Vajna, and Attila Marosi. 2012. Wikipedia as Text. *ERCIM NEWS* 89: 48-49.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[Wang et al.2014] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[Yu and Dredze2014] Mo Yu and Mark Dredze. 2014. Improving Lexical Embeddings with Semantic Knowledge. In *Association for Computational Linguistics (ACL)*.