Semantic Information Extraction for Improved Word Embeddings

Jiaqiang Chen
IIIS, Tsinghua University
Beijing, China
cjq0707@gmail.com

Gerard de Melo
IIIS, Tsinghua University
Beijing, China
gdm@demelo.org

Abstract

Word embeddings have recently proven useful in a number of different applications that deal with natural language. Such embeddings succinctly reflect semantic similarities between words based on their sentence-internal contexts in large corpora. In this paper, we show that information extraction techniques provide valuable additional evidence of semantic relationships that can be exploited when producing word embeddings. We propose a joint model to train word embeddings both on regular context information and on more explicit semantic extractions. The word vectors obtained from such an augmented joint training show improved results on word similarity tasks, suggesting that they can be useful in applications that involve word meanings.

1 Introduction

In recent years, the idea of embedding words in a vector space has gained enormous popularity. This success of such *word embeddings* as semantic representations has been driven in part by the development of novel methods to efficiently train word vectors from large corpora, such that words with similar contexts end up having similar vectors. While it is indisputable that context plays a vital role in meaning acquisition, it seems equally plausible that some contexts would be more helpful for this than others. Consider the following sentence, taken from Wikipedia, a commonly used training corpus for word representation learning:

> Although Roman political authority in the West was lost, Roman culture would last in most parts of the former Western provinces into the 6th century and beyond.

In this example sentence, the token "parts" does not seem to bear any particularly close relationship with the meaning of some of the other tokens, e.g. "Roman" and "culture". In contrast, the occurrence of an expression such as "Greek and Roman mythology" in a corpus appears to indicate that the two tokens "Roman" and "Greek" likely share certain commonalities. There is a large body of work on *information extraction* techniques to discover text patterns that reflect semantic relationships (Hearst, 1992; Tandon and de Melo, 2010).

In this paper, we propose injecting semantic information into word embeddings by training them not just on general contexts but paying special attention to stronger semantic connections that can be discovered in specific contexts on the Web or in corpora. In particular, we investigate mining information of this sort from enumerations and lists, as well as from definitions. Our training procedure can exploit any source of knowledge about pairs of words being strongly coupled to improve over word embeddings trained just on generic corpus contexts.

2 Background and Related Work

Words are substantially discrete in nature, and thus, traditionally, the vast majority of natural language processing tools, both rule-based and statistical,

This research was partially funded by China 973 Program Grants 2011CBA00300, 2011CBA00301, and NSFC Grants 61033001, 61361136003, 20141330245.

have regarded words as distinct atomic symbols. Even methods that rely on vectors typically made use of so-called "one-hot" representations, which allocate a separate dimension in the vector space for every content word in the vocabulary. Such representations suffer from two problems. First, vectors for two distinct word forms have distinct vectors without any overlap, which means that the vector similarities for any two distinct individual word forms will fail to reflect any possible syntactic or semantic similarities between them. Second, the vector space dimensionality is proportional to the vocabulary size, which can be very large. For instance, the Google 1T corpus has 13M distinct words.

To address these two problems, other representations have been proposed. Brown clustering (Brown et al., 1992) organizes words into a binary tree based on the contexts in which they occur. Latent Semantic Analysis and Indexing (LSA/LSI) use singular value decomposition (SVD) to identify the relationships between words in a corpus. Latent Dirichlet Analysis (LDA) (Blei et al., 2003), a generative graphical model, views each document as a collection of topics and assigns each word to these topics.

Recently, neural networks have been applied to learn word embeddings in dense real-valued vector spaces. In training, such an approach may combine vector space semantics with predictions from probabilistic models. For instance, Bengio et al. (2003) present a neural probabilistic language model that uses the n-gram model to learn word embeddings. The network tries to use the first $n-1$ words to predict the next one, outperforming n-gram frequency baselines. Collobert et al. (2011) use word embeddings for traditional NLP tasks: POS tagging, named entity recognition, chunking, and semantic role labeling. Their pairwise ranking approach tries to maximize the difference between scores from text windows in a large training corpus and corresponding randomly generated negative examples. However, the training for this took about one month. The next breakthrough came with Mikolov et al. (2013a), who determined that, for the previous models, most of the complexity is caused by the non-linear hidden layer. The authors thus investigated simpler network architectures to efficiently train the vectors at a much faster rate and thus also at a much larger scale. Their word2vec[1] implementation provides two architectures, the CBOW and the Skip-gram models. CBOW also relies on a window approach, attempting to use the surrounding words to predict the current target word. However, it simplifies the hidden layer to be just the average of surrounding words' embeddings. The Skip-gram model tries to do the opposite. It uses the current word to predict the surrounding words. Both architectures can be trained in just a few hours, while obtaining state-of-the-art embeddings.

Distributed word representations now have been applied to numerous natural language processing tasks. For instance, they have been used for sentiment analysis (Socher et al., 2013), paraphrase detection (Socher et al., 2011), machine translation (Devlin et al., 2014), relation extraction (Chang et al., 2014), and parsing, just to name a few. Some of these works use neural network models, e.g. recursive neural networks, auto-encoders, or convolutional neural networks. Others use word embeddings directly as features for clustering or classification with alternative machine learning algorithms.

There have been other proposals to adapt the word2vec model. Similar to previous work on semantic spaces based on dependency parse relations (Padó and Lapata, 2007), Levy and Goldberg (2014) rely on dependency parsing to create word embeddings. These are able to capture contextual relationships between words that are further apart in the sentence while simultaneously filtering out some words that are not directly related to the target word. Further analysis revealed that their word embeddings capture more functional but less topical similarity. Faruqui et al. (2015) apply post-processing steps to existing word embeddings in order to bring them more in accordance with semantic lexicons such as PPDB and FrameNet. Wang et al. (2014) train embeddings jointly on text and on Freebase, a well-known large knowledge base. Their embeddings are trained to preserve relations between entities in the knowledge graph. Rather than using structured knowledge sources, our work focuses on improving word embeddings using textual data by relying on information extraction to expose particularly valuable contexts in a text corpus.

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

3 Joint Model

Our model simultaneously trains the word embeddings on generic contexts from the corpus on the one hand and semantically significant contexts, obtained using extraction techniques, on the other hand. For the regular general contexts, our approach draws on the word2vec CBOW model (Mikolov et al., 2013a) to predict a word given its surrounding neighbors in the corpus.

At the same time, our model relies on our ability to extract semantically salient contexts that are more indicative of word meanings. Our algorithm assumes that these have been transformed into a set of word pairs known to be closely related. These pairs of related words are used to modify the word embeddings by jointly training them simultaneously with the word2vec model for regular contexts. Due to this more focused information, we expect the final word embeddings to reflect more semantic information than embeddings trained only on regular contexts.

Given an extracted pair of semantically related words, the intuition is that the embeddings for the two words should be pulled together. We formalize this intuition with following objective:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{w_{\mathrm{r}}}\log p(w_{\mathrm{r}}|w_{\mathrm{t}})$$

Here, w_{r} is a word related to another word w_{t} according to the extractions, and T is the vocabulary size.

Thus, given a word w_{t}, we try to maximize the probability of finding its related words w_{r}. Traditionally, the softmax function is used for the probability function. Its time complexity is proportional to the vocabulary size. Here, we use negative sampling as a speed-up technique (Mikolov et al., 2013b). This is a simplified version of Noise Contrastive Estimation (NCE) (Mnih and Teh, 2012), which reduces the problem of determining the softmax to that of binary classification, discriminating between samples from the data distribution and negative samples.

In the training procedure, this amounts to simply generating k random negative samples for each extracted word pair. That is, we replace w_{r} with random words from the vocabulary. For the negative samples, we assign the label $l = 0$, while for the original word pairs, $l = 1$. Now, for each word pair we try to minimize its loss function:

$$Loss = -l \cdot \log f - (1-l) \cdot \log(1-f)$$
$$f = \sigma(v_{w_{\mathrm{t}}}^{T} \cdot v_{w_{\mathrm{r}}})$$

Here, $\sigma(\cdot)$ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and $v_{w_{\mathrm{t}}}$, $v_{w_{\mathrm{r}}}$ refer to the vectors for the two words w_{t} and w_{r}. We use stochastic gradient descent to optimize this function. The formulae for the gradient are easy to compute:

$$\frac{\partial Loss}{\partial v_{w_{\mathrm{r}}}} = -(l-f)\, v_{w_{\mathrm{t}}}$$
$$\frac{\partial Loss}{\partial v_{w_{\mathrm{t}}}} = -(l-f)\, v_{w_{\mathrm{r}}}$$

This objective is optimized alongside with the original word2vec CBOW objective. Our overall model combines the two objectives. Training the model in parallel with the word2vec model allows us to inject the extracted knowledge into the word vectors such that they are reflected during the word2vec training rather than just as a post-processing step. Thus the two components are able to mutually influence each other.

Both objectives contribute to the embeddings' ability to capture semantic relationships. Training with the extracted contexts enables us to adjust word embeddings based on concrete evidence of semantic relationships, while the use of general corpus contexts enables us to maintain the advantages of the word2vec model, in particular its ability to benefit from massive volumes of raw corpus data.

4 Information Extraction

Our model can flexibly incorporate semantic relationships extracted using various kinds of information extraction methods. Different kinds of sources and extraction methods can bring different sorts of information to the vectors, suitable for different applications. In our experiments, we investigate two sources: a dictionary corpus from which we extract definitions and synonyms, and a general Web corpus, from which we extract lists. Our model could similarly be used with other extraction methods, or in fact any method to mine pairs of semantically related words.

word w_t	**definition** w_r **for** w_t
Befuddle	to becloud and confuse as with liquor
Befuddled	dazed by alcoholic drink
Befuddled	unclear in mind or intent filled with bewilderment
Befuddled	confused and vague, used especially of thinking
Beg	to ask earnestly for, to entreat, or supplicate for, to beseech
word w_t	**synonym** w_r **for** w_t
Effectual	effectual, efficacious, effective
Effectuality	effectiveness, effectivity, effectualness
Efficacious	effectual
Efficaciousness	efficacy

Table 1: Definitions and synonyms from the GCIDE

player, captain, manager, director, vice-chairman
group, race, culture, religion, organisation, person
Italian, Mexican, Chinese, Creole, French
prehistoric, roman, early-medieval, late-medieval, post-medieval, modern
ballscrews, leadscrews, worm gear, screwjacks, linear, actuator
Cleveland, Essex, Lincolnshire, Northamptonshire, Nottinghamshire, Thames Valley, South Wales

Table 2: Lists of related words extracted from UKWaC

4.1 Definition Extraction

One can safely assume that any large, broad-coverage Web corpus will contain significant occurrences of word definitions, e.g. whenever new terminology is introduced. These can be harvested using broad-coverage Definition Extraction methods (Sierra et al., 2009).

Instead of adopting such generic methods that are intended to operate on arbitrary text, another option, appropriate for Web corpora, is to specifically identify the kinds of Web sources that provide high-quality definitions, e.g. Wiktionary or Wikipedia. In fact, when compiling a Web corpus with the explicit purpose of using it to train word representations, one may reasonably wish to explicitly ensure that it includes dictionaries available on the Web. Obviously, the definitions from a dictionary can provide meaningful semantic relationships between words.

In our experiments, we use the GNU Collaborative International Dictionary of English (GCIDE) as our dictionary corpus, which is derived from an older edition of Webster's Revised Unabridged Dictionary. From this data, we extract the dictionary glosses as genuine definitions as well as synonyms. In the dictionary, they are indexed by the $\langle def \rangle$ and $\langle syn \rangle$ tags. We ignore other embedded tags within the definitions and synonym entries. These provide additional word usage notes and other attributes that are not significant in our work. In total, we obtain 208,881 definition entries. Some words have multiple meanings and thus are part of several entries. We also obtain 10,148 synonym entries, each of which consists of one or more synonyms for a given word. Table 1 shows some examples of this extraction. We can observe that the definition and synonym extractions indeed appear to convey valuable information about semantic proximity of words.

4.2 List Extraction

Lists and enumerations are another promising source of information. Words that occur together within a list are not just semantically connected but often even of the same type. These sorts of contexts thus also have the potential to improve the word embeddings. We extract them from the UKWaC corpus (Baroni et al., 2009), a general broad-coverage corpus crawled from the Web, but limited to the `.uk` domain. After post-crawl cleaning, it contains a total of about 2 billion words. It is annotated with POS and dependency tags, which simplifies our work of extracting high-quality lists.

To extract lists of similar words, we use a simple rule-based method. We first search for continuous appearances of commas, which indicate a possible list of similar things. To filter out noise, we require that the entries in the list be approximately of equal length. The length of each entry should be in the range from 1 to 4. Longer entries are much more likely to be short sentences or clauses, which are not very useful when our aim is to obtain lists of similar words. We also restrict list items to be nouns and adjectives using the POS tags provided with the UKWaC.

Additionally, we rely on special search patterns matching for instance *"include A, B, C, (and) D"*, *"A and(or) B"*, *"cities(or other nouns) like A, B, C, D"*, *"cities(or other nouns) such as A, B, C, (and) D"*, etc. Here, the letters *A*, *B*, and so on, refer to the extracted target words, while other words and punctuation, merely indicating the occurrence of the lists, are removed, e.g. commas or the word *"and"*.

In total, 339,111 lists are extracted from the UKWaC, examples of which are shown in Table 2. We see that although there is some noise, the list extraction also captures semantically meaningful relationships. The words in the lists tend to be of the same or similar type and represent similar or related things.

5 Experiments and Evaluation

In order to investigate the impact of extractions on word embeddings, we conduct an empirical analysis based on semantic relatedness assessments.

5.1 Data

Our model relies on two types of input. For semantically salient contexts, we rely on the data and extraction techniques described above in Section 4 to obtain pairs of related words.

For the regular contexts used by the CBOW model, we rely on a 2010 Wikipedia data set[2]. We normalize the text to lower case and remove special characters. After prepocessing, it contains 1,205,009,210 tokens. We select words appearing at least 50 times and obtain a vocabulary of 220,521 words.

[2]`http://nlp.stanford.edu/data/WestburyLab.wikicorp.201004.txt.bz2`

5.2 Training

Having obtained two kinds of extracted word contexts, we use these separately to train word embeddings jointly with the word2vec model. Our training implementation relies on a multi-threaded architecture in which some threads optimize for the original word2vec objective, training on different parts of the corpus. At the same time, alongside with these threads, further threads optimize based on the extracted pairs of words using the objective given earlier. All threads asynchronously update the word embeddings, using stochastic gradient descent steps. Thus, both the raw corpus for the word2vec model and the related word pairs can bring their information to bear on the word embeddings.

We use 20 threads for the CBOW architecture, which runs faster than the Skip-gram model. The window size of the CBOW model is set to 8. We run it for 3 passes over the Wikipedia data set, which is sufficient to achieve good results. We sample 10 random words as negative examples for each instance.

Additional threads are used for the extracted pairs of words. We use 4 threads each for lists and definitions (by splitting definitions) and one thread for synonyms. In each case, the extractions lead to positive pairs of semantically related words. For definitions and synonyms, the word pair consists of a headword and one word from its definition, or of the headword and one of its synonyms. For the list extraction setup, the training word pairs consist of any two words from the same list. For these word pairs, we also randomly sample 10 words as the corresponding negative examples. They update the word embeddings jointly with the CBOW model. This way, the semantic information they contain can be used to adjust the results from word2vec.

We use different learning rates to control each source's contribution to the final word embeddings. We set the initial learning rate for the CBOW threads to be 0.050 and report results for different rates for the other threads, ranging from 0.001 to 0.1.

We stop training the word embeddings in the fol-

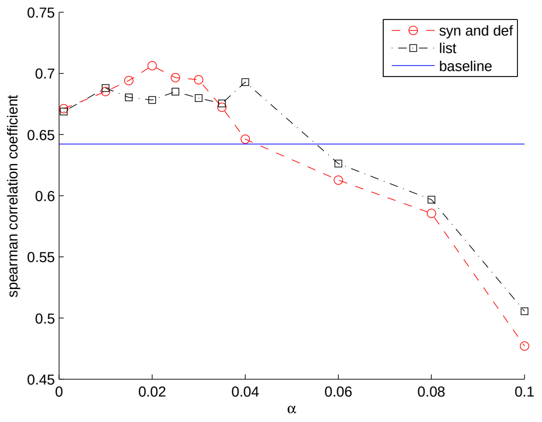

Figure 1: Spearman's ρ for wordsim-353

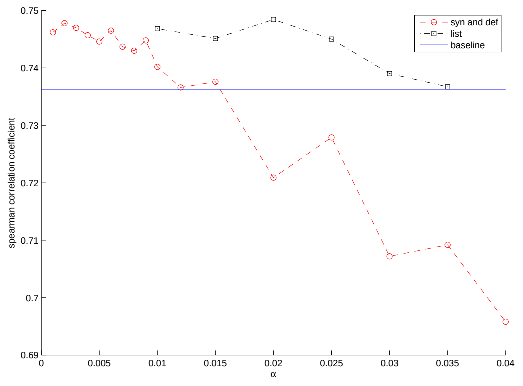

Figure 3: Spearman's ρ for MEN

lowing way to ensure convergence: when all the CBOW threads finish, the other threads are terminated. This is because the extractions are supplementary to the CBOW model, which is the main source to train the word vectors. This mode of operation also ensures that we are always training on both components of the model jointly rather than allowing one component to dominate towards the end. We did also experiment with pre-defined numbers of iterations for the additional threads to control convergence, but the results were not very different.

5.3 Evaluation and Analysis

We use the wordsim-353 (Finkelstein et al., 2001) and MEN (Bruni et al., 2014) datasets to evaluate the semantic similarities reflected in the final word embeddings. Wordsim-353 and MEN are datasets of English word pairs with human-assigned similarity judgements. They are often used to train or test semantic similarity measures of words. We calculate the cosine distance of word embeddings for the word pairs in wordsim-353 and MEN and compare them to the scores from human annotations.

Fig. 1 shows the Spearman's correlation coefficients for the wordsim-353 dataset. Even for a learning rate α as low as 0.001 for the additional threads, we can obtain some improvement over the CBOW baseline (which corresponds to an α setting of 0.0). As α increases, the result gets better. The best result we get for synonyms and definitions is 0.706, while for lists from UKWaC, it is 0.693. The best learning rate for the definitions and synonyms is 0.020, while for the list extractions, it is 0.040. Both lead to noticeably better results than the CBOW baseline's correlation coefficient of 0.642. Note that for large α, the augmentation performs worse than the baseline. This is expected, as an overly high learning rate causes information from the related words to overwhelm the original CBOW model, leading to excessively biased final embeddings.

Fig. 3 plots the results on the MEN dataset. The best-performing learning rate is different from that for wordsim-353. In particular, well-performing learning rates are slightly smaller. For the definitions and synonyms, the best is 0.002, while for the lists, the best learning rate is 0.020. After training jointly, we obtain higher correlation coefficients for the MEN similarity task.

	wordsim-353	MEN
CBOW (baseline)	0.642	0.736
Definitions and synonyms	0.706	0.748
Lists	0.693	0.749

Table 3: Best results for the word similarity tasks

Table 3 provides a summary of the best results that we obtain on the two similarity tasks.[3] It can be seen that we obtain higher correlation coefficients for these tasks. This suggests that the word vectors

[3]Unfortunately, there is no independent tuning set from the same distribution and thus we follow previous work in reporting best results on the final set.

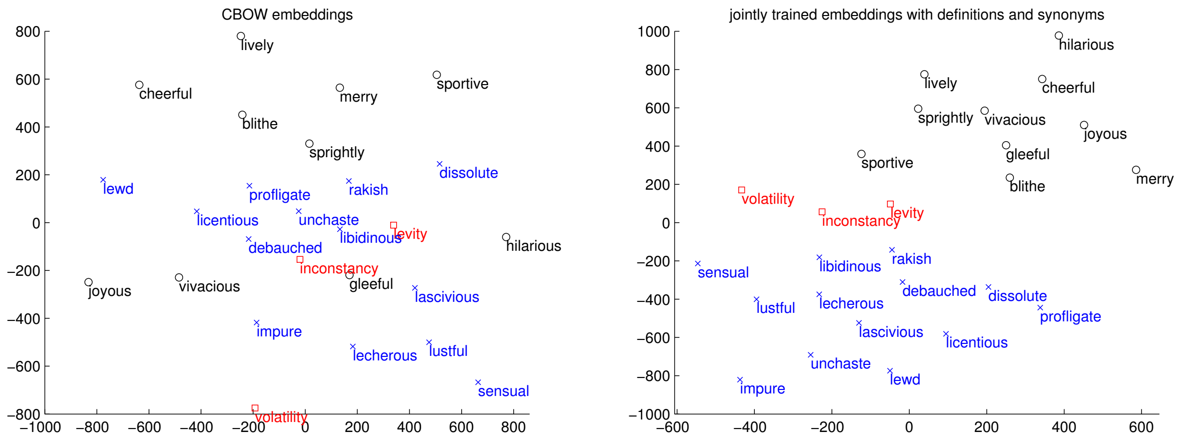

Figure 2: A t-SNE visualization of the word embeddings, comparing the CBOW baseline (left) with that of our joint model (right)

capture more semantic properties of words and thus may be used in applications that benefit from semantic information.

Finally, we plot a sample of the word embeddings obtained from the joint training with definitions and synonyms using t-SNE (van der Maaten and Hinton, 2008). t-SNE is a technique for visualization of high-dimensional datasets using dimensionality reduction. The perplexity of the Gaussian kernel for the t-SNE is set to 15. Figure 2 shows a plot for 26 words: *levity*, *lewd*, and *merry*, and their synonyms. We see that our model successfully distinguishes the different meanings of these words while reflecting semantic relationships.

6 Conclusion

We have presented a way to improve word embeddings by drawing on the idea that certain contexts exhibit more semantically meaningful information than others. Information extraction methods allow us to discover such contexts and mine semantic relationships between words. We focus on word definitions and synonyms, as well as on lists and enumerations. The final word embeddings after joint training show better correlation coefficients in similarity tasks. This suggests that information extraction methods can help word vectors capture more meaning, making them useful for semantic applications and calling for further research in this area.

References

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.*, 49(1):1–47, January.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1579. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the*

52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1370–1380, Baltimore, Maryland, June. Association for Computational Linguistics.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Lev Finkelstein, Gabrilovich Evgenly, Matias Yossi, Rivlin Ehud, Solan Zach, Wolfman Gadi, and Ruppin Eytan. 2001. Placing search in context: the concept revisited. In *Proceedings of the Tenth International World Wide Web Conference*.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545.

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: Short Papers)*, pages 302–308. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758.

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

Gerardo Sierra, Maria Pozzi, and Juan-Manuel Torres, editors. 2009. *WDE '09: Proceedings of the 1st Workshop on Definition Extraction*, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NIPS*, pages 801–809.

Richard Socher, Alex Perelygin, Jean 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 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October. Association for Computational Linguistics.

Niket Tandon and Gerard de Melo. 2010. Information extraction from web-scale n-gram data. In Chengxiang Zhai, David Yarowsky, Evelyne Viegas, Kuansan Wang, and Stephan Vogel, editors, *Web N-gram Workshop. Workshop of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, volume 5803, pages 8–15. ACM.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601, Doha, Qatar, October. Association for Computational Linguistics.