Improving Word and Sense Embedding with Hierarchical Semantic Relations

Yow-Ting Shiue
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
orina1123@gmail.com

Wei-Yun Ma
Institute of Information Science
Academia Sinica
Taipei, Taiwan
ma@iis.sinica.edu.tw

***Abstract*—Distributed representations are shown useful in many NLP tasks. Besides the context, lexical resources also provide valuable information about lexical units. This paper proposes simple methods to improve word and sense vectors by training on multi-level hierarchical semantic relations. Experiments on both intrinsic and extrinsic tasks show that our approach leads to consistent improvement competitive with state-of-the-arts. Moreover, the enhanced vectors are directly applicable to existing applications.**

***Keywords*-word embedding; sense embedding; distributed representation; WordNet; semantic relation**

I. Introduction

Distributed word representations have been widely used in various NLP tasks. Many methods have been proposed to learn word embedding using large corpus [1], [2], [3], [4], [5]. To improve the quality of context-based embeddings, researchers have explored to make full use of lexical resources such as WordNet [6]. [7] modify the objective to include knowledge about synonyms. [8] retrofit word embeddings by encouraging linked words to have similar representations, but only synonyms and "direct" (single-level) hypernyms/hyponyms are considered. [9] utilize semantic knowledge including synonym, antonym and hyponym/hyernym, and formulate a constrained optimization problem. However, the strength of constraints does not vary with the number of levels the hyponym and hypernym are across.

Another line of research aims to produce sense-specific embeddings. AutoExtend [10] learns synset embeddings with WordNet relations as constraints. SensEmbed [11] uses sense annotated text to train sense vectors and utilize relations in a semantic network called BabelNet [12] when computing word similarities. However, these relations were not used to update the vectors during training, so the gains from the lexical resource could not be directly transfered to practical applications. In both methods, multi-level hyponym-hypernym relations were not considered. Neither did they provide a way to obtain better word vectors based on sense vectors.

In this paper, to better utilize hyponym-hypernyms relations, we propose two new approaches for learning sense and word embeddings. The first one improves context-based sense embeddings by pre-training or post-processing on such relations. The second one maps the sense embeddings trained on relations to word embeddings and continues to train them on an ordinary corpus. We model the importance of a relation by a function of its distance. The main idea is that all hyponym-hypernym relations, including multi-level ones, are supposed to be used and should be used differentially according to their distances.

II. Methods

A. Sense Vectors

The hyponym-hypernym relations, which we referred to as hierarchical relations, reflects an organized hierarchy of concepts in the language. In this paper we treat the relations as undirected ones and evaluate mainly with the word similarity task, which is symmetric by definition.

We dumped 766,158 hierarchical relations from WordNet 3.0, including direct and multi-level ones, and use a file containing the relation pairs one per line as the input to the Word2vec Skip-gram (SG) model.

Handling the distance factor: Hierarchical relations can have different distances. For instance, "cherry" and "tree" are close in the WordNet hierarchy, but "cherry" and "object" have a longer distance. The closer the two senses are, the larger impact they should have on each other during training. We use a linear function to reflect the level of impact. Let $d(s_i, s_j)$ be the distance or shortest path length of hyponym-hypernym senses s_i and s_j in the WordNet graph of synsets. The weight of a pair (s_1, s_2) is: $weight(s_1, s_2) = \max_{i,j} d(s_i, s_j) - d(s_1, s_2) + 1$ where the last term prevents zero weights.

We experimented two approaches to incorporate this weight factor into the SG model.
(1) ***wn_cnt***: We let the relation pair (s_1, s_2) occur $weight(s_1, s_2)$ times in the training file.
(2) ***wn_dis***: When using the vector of sense s_1 to update that of sense s_2, or vice versa, multiply the gradient by $weight(s_1, s_2)$.

The version without any weighting on relations is denoted as ***wn_all***. To verify the usefulness of multi-level relations, we implement a version called ***wn_dir*** in which only direct ($d(s_1, s_2) = 1$) relation pairs are used.

Pre-training v.s. post-processing: To leverage hierarchical relations, we consider initializing the sense vectors with the vectors pre-trained on the relations (*wn_all*, *wn_cnt*, *wn_dis*). We save both target and context vectors for initialization. The initialized vectors can then be trained on a sense-annotated corpus. The motivation behind this design lies in the difference of characteristics between semantic relation data and corpus data. The relation data provides fairly accurate relationship between concepts but is rather sparse, while the corpus data is very dense and describes a variety of possible types of connection among words. By pre-training on the hierarchical relations, we aim to build a framework of core concepts into the model, so that it may learn better in the subsequent corpus training phase. For a comparison, we also experimented a version in which the sense vectors trained with corpus are post-processed with WordNet relations.

B. Word Vectors

Although [11] has shown the benefits of moving from word to sense vectors, performing word sense disambiguation (WSD) may not be practical in all applications. When dealing with very large text data, WSD can be overly time-consuming. Therefore, we also propose an approach to obtaining relation-enhanced word vectors. We design two methods for mapping pre-trained sense vectors to word vectors:
(1) ***First sense (FS)***: For each word w, we assign the vector of its first sense (predominant sense) $s_{w,1}$ to its initial vector.
(2) ***Weighted senses (WS)***: Let $freq(w, s_{w,i})$ be the number of times word w is associated with sense $s_{w,i}$ in a disambiguated corpus. The initial vector of word w with n possible senses is:

$$vec(w) = \frac{\sum_{i=1}^{n} freq(w, s_{w,i})\ vec(s_{w,i})}{\sum_{i=1}^{n} freq(w, s_{w,i})}$$

We can then use an ordinary corpus to train the initialized word vectors. One advantage of this mapping approach is that there is no need to expand a sense-level relation (s_1, s_2) into $size(s_1) * size(s_2)$ word-level ones, which will results in overwhelmingly large number of relations since we are including multi-level hierarchical relations.

III. Experimental Results and Analysis

In the experiments, we mainly follow the choices of [11]. The dimensionality is 400, the window size is 5, and the number of negative samples is 15. We use the SimLex-999 [13] word similarity dataset as a validation set to select one best model for each method among different number of iterations. To also report the performance on SimLex-999, we use WS-353 [14] for validation.

We use the December-2015 dump of the English Wikipedia as our training corpus. The data is converted to plaintext and then tokenized with CoreNLP Document-Preprocessor[1]. To obtain a sense-annotated corpus, the pywsd[2] implementation of Adapted Lesk algorithm [15] is adopted.

A. Sense Vectors

We use the Word2vec SG model trained only on the sense-annotated corpus as the baseline. We also compare with the result of SensEmbed reported by [11] and the result of pre-trained AutoExtend synset embeddings.[3]

To adopt sense vectors to compute word similarities, we use two methods.
(1) ***closest***: The similarity of two words is defined to be the similarity of their closest senses.
(2) ***weighted***: We compose the vector of a word by summing the vectors of all its possible senses, weighted by the word-sense frequency in the disambiguated corpus.[4]

Rationale of using hierarchical relations: To show the usefulness of hierarchical relations, we demonstrate the performance of SG trained solely on relation without any corpus data. The results on the SimLex-999 dataset with the *closest* measurement are shown in Table I. The increasing performance of *wn_all* and *wn_dis* with more number of iterations shows that it is possible to learn word similarity information from hierarchical relations. Using multi-level relations is better than using only direct relations (*wn_dir*). The best result of *wn_cnt* and *wn_dis* are both higher than the best of *wn_all*, showing that a linear function of the relation distance can help.

Word similarity tasks: We evaluate the relation-enhanced sense vectors on standard word similarity/relatedness datasets: RG-65 [16], WS-353, WS-SIM, WS-REL [17], YP-130 [18], MEN [19] and SimLex-999. According to Table II (*Full Corpus* part), under the *closest* measurement, pre-training leads to some improvement. Under *weighted*, the models post-processed with *wn_dis* outperforms the baseline on RG-65, WS-353 and YP-130. However, post-processing lowers the performance in some cases so application-specific tuning is required. In general, it is more important to consider relation weights in post-processing than in pre-training, possibly because the noise introduced by long-distance relations could be "corrected" when training on the corpus.

The difference between the proposed methods and the baseline is slight, probably because the amount of relation data is too small compared to the corpus. Therefore, we use 10% of the disambiguated corpus to conduct the experiments and report the results in the left part of Table II. Due to space limit we only include two datasets on which the difference is less obvious in the *Full Corpus* experiments. For pre-training, *wn_cnt* is preferred to *wn_dis*, since in the initialization step the vectors are still rather random and it is less meaningful to update

# iter	wn_dir	wn_all	wn_cnt	wn_dis
Rand.	0.053			
5	0.054	0.117	0.319	0.123
10	0.039	0.133	**0.366**	0.136
50	0.129	0.226	0.355	0.243
100	0.127	0.284	0.328	0.306

Table I: Performance on the SimLex-999 dataset of the sense models trained solely on hierarchical relations (*closest*, Spearman correlation).

[1]http://nlp.stanford.edu/software/tokenizer.shtml
[2]https://github.com/alvations/pywsd
[3]Both results are not directly comparable to ours due to different training corpus, parameters and sense systems.

[4]Note that this is different from the "weighted" measurement used by [11], which weighs on the similarities instead of vectors. In our pilot experiments, our method leads to consistently higher correlation coefficient.

Similarity	**Method**		*10% Corpus*		*Full Corpus*						
			WS353	**MEN**	**RG65**	**WS353**	**WS-S**	**WS-R**	**YP130**	**MEN**	**SL999**
closest	SG Baseline		0.674	0.727	**0.836**	0.673	0.780	**0.556**	0.747	**0.761**	0.444
	Pre	wn_all	**0.677**	0.730	0.809	**0.675**	**0.785**	0.549	**0.759**	**0.761**	**0.448**
		wn_cnt	**0.677**	**0.731**	0.812	0.673	0.783	0.543	0.742	0.749	0.441
		wn_dis	0.668	0.729	0.805	0.674	0.783	0.538	0.740	0.745	0.446
	Post	wn_all	0.404	0.357	0.802	0.538	0.673	0.302	0.677	0.563	0.434
		wn_cnt	0.390	0.361	0.811	0.525	0.684	0.295	0.649	0.570	0.443
		wn_dis	0.400	0.355	0.813	0.557	0.680	0.380	0.612	0.579	0.387
	SensEmbed *		-	-	*0.825*	-	*0.693*	*0.488*	*0.492*	*0.712*	-
	AutoExtend *		-	-	*0.867*	*0.557*	*0.730*	*0.387*	*0.695*	*0.747*	*0.517*
weighted	SG Baseline		0.672	0.735	0.822	0.697	0.777	**0.609**	0.621	0.767	0.416
	Pre	wn_all	0.678	**0.747**	0.819	0.691	0.773	0.589	0.614	0.765	0.425
		wn_cnt	**0.690**	0.743	0.833	0.689	0.770	0.590	0.612	**0.768**	0.417
		wn_dis	0.675	0.732	0.827	**0.699**	**0.778**	0.605	0.616	0.766	**0.430**
	Post	wn_all	0.654	0.644	0.835	0.685	0.762	0.561	**0.609**	0.742	0.405
		wn_cnt	0.632	0.608	0.826	0.646	0.739	0.480	0.615	0.704	0.397
		wn_dis	0.653	0.643	**0.841**	**0.699**	0.760	0.585	**0.633**	0.738	0.404
	SensEmbed *		-	-	*0.877*	-	*0.776*	*0.639*	*0.446*	*0.783*	-
	AutoExtend *		-	-	*0.730*	*0.605*	*0.721*	*0.464*	*0.664*	*0.732*	*0.463*

Table II: Word similarity performance of sense models (Spearman correlation). *: different settings.

Method		*10% Corpus*		*Full Corpus*								
		WS353	**MEN**	**RG65**	**WS353**	**WS-S**	**WS-R**	**YP130**	**MEN**	**SL999**	**SA**	**Dependency**
SG Baseline		0.686	0.672	0.777	0.720	0.798	0.644	0.501	0.759	0.398	0.761	85.90 / 82.62
wn_all	FS	0.695	0.712	0.804	**0.729**	**0.804**	0.656	0.519	0.762	0.401	0.760	86.60 / 83.39
	WS	**0.717**	0.715	0.809	0.724	0.802	0.650	0.510	0.762	0.405	0.765	86.29 / 83.15
wn_cnt	FS	0.697	0.704	**0.829**	0.720	0.797	0.644	0.500	0.761	0.406	0.764	86.74 / 83.68
	WS	**0.717**	**0.726**	0.825	0.725	0.793	0.650	0.510	**0.767**	0.400	0.759	**87.82 / 84.92**
wn_dis	FS	0.700	0.718	0.822	0.724	0.802	**0.664**	0.516	0.756	0.409	0.762	85.93 / 82.71
	WS	0.693	0.713	0.805	0.724	0.787	0.657	0.526	0.762	0.405	0.764	87.20 / 84.26
Retro(WN$_{all}$)		0.615	0.588	0.827	0.670	0.788	0.556	**0.551**	0.711	**0.475**	**0.770**	85.97 / 82.77

Table III: Performance of word models on word similarity (Spearman correlation), sentiment analysis (accuracy), and dependency parsing (UAS/LAS).

one with another with a large weight. Post-processing can degrade the performance a lot when there is no sufficient corpus data.

B. Word Vectors

For this set of experiments, we obtain relation-enhanced word vectors by taking the following steps: 1) take the model trained solely on relations; 2) map the sense vectors to word vectors (by *FS* or *WS*); 3) continue to train the word vectors on Wikipedia. We compare with the retrofitting method [8], using the released implementation to process our SG baseline model with the WN$_{all}$ setting, which includes synonyms and direct hypernyms/hyponyms.

Word similarity tasks: We evaluate our word vectors on the word similarity datasets and report the results in Table III. As can be seen, our initialization method leads to consistent improvement across different datasets. Retrofitting, in contrast, performs rather unstably, for which dramatic drop in performance can be observed on WS-353, WS-REL and MEN. This might be explained by the fact that closely-related concepts could be far apart in the WordNet hierarchy. For example, in WS-REL, the shortest distance on WordNet between (country, citizen) is 8. Multi-level relations help to capture more connections among words. In fact, our *wn_dis* pre-training with *FS* mapping model does rank this pair higher than (computer, news), whose shortest distance is 10. The two pairs are ranked reversely by the baseline model and retrofitting, which does not use multi-level relations.

We also conduct experiments with 10% of the corpus and show the result in the left part of Table III. With less corpus data, more obvious improvement over the baseline can be achieved by our methods. This indicates that our method can be applied to some domain-specific tasks. For example, in the biomedical domain, there might be existing ontologies of terms but less available text. In contrast, retrofitting does not help on the two datasets when the corpus data is small.

Sentiment analysis: We use the movie review dataset provided by [20] and train a logistic regression classifier with l2-regularization for binary classification (positive/negative). The feature of each sentence is the average of the embeddings of all words in it. As can be seen in Table III, most of our enhanced word vectors can serve as better features and help the classifier to achieve higher performance, but not as effective as retrofitting.

Dependency parsing: We use different embeddings for input features in the Stanford Neural Network Dependency Parser [21] and report unlabeled (UAS) and labeled attachment scores (LAS) on the test set of English Penn Treebank (PTB). We set the embedding size to 400 and use the default for other parameters. Under this setting, the model with random initial embeddings achieves UAS = 85.58 and LAS = 82.42. As can be seen in Table III, all of our methods outperform the SG baseline, and *wn_cnt* pre-training with *WS* mapping performs the best. The improvement might be attributed to the inclusion of ontological knowledge, which can serve as a clue for determining whether there is a dependency between two words.

IV. Conclusion

In this paper we propose simple but effective methods of utilizing hierarchical semantic relations to improve sense and word vectors. We model the importance of a relation according to its distance. Our results are competitive with state-of-the-art methods, and the consistent improvement on intrinsic and extrinsic evaluations shows that our enhanced vectors are directly applicable to existing applications. When the amount of corpus data is limited, our methods are especially useful, and pre-training is more reliable than post-processing in such cases.

In the future, it is worth investigating how to encode directional information. For example, "cherry" is a kind of "tree" so it should inherit the properties of "tree", but some properties of "cherry" might not apply to all kinds of "tree".

References

[1] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa, "Natural language processing (almost) from scratch," *CoRR*, vol. abs/1103.0398, 2011.

[2] P. S. Dhillon, J. Rodu, D. P. Foster, and L. H. Ungar, "Using CCA to improve CCA: A new spectral method for estimating vector models of words," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, 2013, pp. 3111–3119.

[4] R. Lebret and R. Collobert, "Word embeddings through hellinger PCA," in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, 2014, pp. 482–490.

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 2014, pp. 1532–1543.

[6] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[7] M. Yu and M. Dredze, "Improving lexical embeddings with semantic knowledge," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, 2014, pp. 545–550.

[8] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," *arXiv preprint arXiv:1411.4166*, 2014.

[9] Q. Liu, H. Jiang, S. Wei, Z.-H. Ling, and Y. Hu, "Learning semantic word embeddings based on ordinal knowledge constraints," in *ACL*, 2015.

[10] S. Rothe and H. Schütze, "Autoextend: Extending word embeddings to embeddings for synsets and lexemes," *arXiv preprint arXiv:1507.01127*, 2015.

[11] I. Iacobacci, M. T. Pilehvar, and R. Navigli, "Sensembed: learning sense embeddings for word and relational similarity," in *Proceedings of ACL*, 2015, pp. 95–105.

[12] R. Navigli and S. P. Ponzetto, "Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network," *Artificial Intelligence*, vol. 193, pp. 217–250, 2012.

[13] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, 2016.

[14] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 406–414.

[15] S. Banerjee and T. Pedersen, "An adapted lesk algorithm for word sense disambiguation using wordnet," in *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 2002, pp. 136–145.

[16] H. Rubenstein and J. B. Goodenough, "Contextual correlates of synonymy," *Communications of the ACM*, vol. 8, no. 10, pp. 627–633, 1965.

[17] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa, "A study on similarity and relatedness using distributional and wordnet-based approaches," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2009, pp. 19–27.

[18] D. Yang and D. M. Powers, "Measuring semantic similarity in the taxonomy of wordnet," in *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*. Australian Computer Society, Inc., 2005, pp. 315–322.

[19] E. Bruni, N.-K. Tran, and M. Baroni, "Multimodal distributional semantics." *J. Artif. Intell. Res.(JAIR)*, vol. 49, no. 1-47, 2014.

[20] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631. Citeseer, 2013, p. 1642.

[21] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks." in *EMNLP*, 2014, pp. 740–750.