Improving Lexical Embeddings with Semantic Knowledge

Mo Yu [*]
Machine Translation Lab
Harbin Institute of Technology
Harbin, China
gflfof@gmail.com

Mark Dredze
Human Language Technology Center of Excellence
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218
mdredze@cs.jhu.edu

Abstract

Word embeddings learned on unlabeled data are a popular tool in semantics, but may not capture the desired semantics. We propose a new learning objective that incorporates both a neural language model objective (Mikolov et al., 2013) and prior knowledge from semantic resources to learn improved lexical semantic embeddings. We demonstrate that our embeddings improve over those learned solely on raw text in three settings: language modeling, measuring semantic similarity, and predicting human judgements.

1 Introduction

Word embeddings are popular representations for syntax (Turian et al., 2010; Collobert and Weston, 2008; Mnih and Hinton, 2007), semantics (Huang et al., 2012; Socher et al., 2013), morphology (Luong et al., 2013) and other areas. A long line of embeddings work, such as LSA and randomized embeddings (Ravichandran et al., 2005; Van Durme and Lall, 2010), has recently turned to neural language models (Bengio et al., 2006; Collobert and Weston, 2008; Turian et al., 2010). Unsupervised learning can take advantage of large corpora, which can produce impressive results.

However, the main drawback of unsupervised learning is that the learned embeddings may not be suited for the task of interest. Consider semantic embeddings, which may capture a notion of semantics that improves one semantic task but harms another. Controlling this behavior is challenging with an unsupervised objective. However, rich prior knowledge exists for many tasks, and there are numerous such semantic resources.

We propose a new training objective for learning word embeddings that incorporates prior

---

This work was done while the author was visiting JHU.

knowledge. Our model builds on word2vec (Mikolov et al., 2013), a neural network based language model that learns word embeddings by maximizing the probability of raw text. We extend the objective to include prior knowledge about synonyms from semantic resources; we consider both the Paraphrase Database (Ganitkevitch et al., 2013) and WordNet (Fellbaum, 1999), which annotate semantic relatedness between words. The latter was also used in (Bordes et al., 2012) for training a network for predicting synset relation. The combined objective maximizes both the probability of the raw corpus and encourages embeddings to capture semantic relations from the resources. We demonstrate improvements in our embeddings on three tasks: language modeling, measuring word similarity, and predicting human judgements on word pairs.

2 Learning Embeddings

We present a general model for learning word embeddings that incorporates prior knowledge available for a domain. While in this work we consider semantics, our model could incorporate prior knowledge from many types of resources. We begin by reviewing the word2vec objective and then present augmentations of the objective for prior knowledge, including different training strategies.

2.1 Word2vec

Word2vec (Mikolov et al., 2013) is an algorithm for learning embeddings using a neural language model. Embeddings are represented by a set of latent (hidden) variables, and each word is represented by a specific instantiation of these variables. Training learns these representations for each word w_t (the tth word in a corpus of size T) so as to maximize the log likelihood of each token given its context: words within a window sized c:

$$\max \frac{1}{T} \sum_{t=1}^{T} \log p\left(w_t | w_{t-c}^{t+c}\right), \qquad (1)$$

where w_{t-c}^{t+c} is the set of words in the window of size c centered at w_t (w_t excluded).

Word2vec offers two choices for modeling of Eq. (1): a skip-gram model and a continuous bag-of-words model (cbow). The latter worked better in our experiments so we focus on it in our presentation. cbow defines $p(w_t|w_{t-c}^{t+c})$ as:

$$\frac{\exp\left({e'_{w_t}}^\top \cdot \sum_{-c \le j \le c, j \neq 0} e_{w_{t+j}}\right)}{\sum_w \exp\left({e'_w}^\top \cdot \sum_{-c \le j \le c, j \neq 0} e_{w_{t+j}}\right)}, \quad (2)$$

where e_w and e'_w represent the input and output embeddings respectively, i.e., the assignments to the latent variables for word w. While some learn a single representation for each word ($e'_w \triangleq e_w$), our results improved when we used a separate embedding for input and output in cbow.

2.2 Relation Constrained Model

Suppose we have a resource that indicates relations between words. In the case of semantics, we could have a resource that encodes semantic similarity between words. Based on this resource, we learn embeddings that predict one word from another related word. We define $\mathbf{R}$ as a set of relations between two words w and w'. $\mathbf{R}$ can contain typed relations (e.g., w is related to w' through a specific type of semantic relation), and relations can have associated scores indicating their strength. We assume a single relation type of uniform strength, though it is straightforward to include additional characteristics into the objective.

Define $\mathbf{R}_w$ to be the subset of relations in $\mathbf{R}$ which involve word w. Our objective maximizes the (log) probability of all relations by summing over all words N in the vocabulary:

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{w \in \mathbf{R}_{w_i}} \log p\left(w|w_i\right), \quad (3)$$

$p(w|w_i) = \exp\left({e'_w}^T e_{w_i}\right)/\sum_{\bar{w}} \exp\left({e'_{\bar{w}}}^T e_{w_i}\right)$
takes a form similar to Eq. (2) but without the context: e and e' are again the input and output embeddings. For our semantic relations e'_w and e_w are symmetrical, so we use a single embedding. Embeddings are learned such that they are predictive of related words in the resource. We call this the Relation Constrained Model (RCM).

2.3 Joint Model

The cbow and RCM objectives use separate data for learning. While RCM learns embeddings

suited to specific tasks based on knowledge resources, cbow learns embeddings for words not included in the resource but appear in a corpus. We form a joint model through a linear combination of the two (weighted by C):

$$\frac{1}{T} \sum_{t=1}^{T} \log p\left(w_t|w_{t-c}^{t+c}\right) + \frac{C}{N} \sum_{i=1}^{N} \sum_{w \in \mathbf{R}_{w_i}} \log p\left(w|w_i\right)$$

Based on our initial experiments, RCM uses the output embeddings of cbow.

We learn embeddings using stochastic gradient ascent. Updates for the first term for e' and e are:

$$e'_w - \alpha_{\text{cbow}} \left(\sigma(f(w)) - I_{[w=w_t]}\right) \cdot \sum_{j=t-c}^{t+c} e_{w_j}$$

$$e_{w_j} - \alpha_{\text{cbow}} \sum_{w} \left(\sigma(f(w)) - I_{[w=w_t]}\right) \cdot e'_w,$$

where $\sigma(x) = \exp\{x\}/(1 + \exp\{x\})$, $I_{[x]}$ is 1 when x is true, $f(w) = {e'_w}^\top \sum_{j=t-c}^{t+c} e_{w_j}$. Second term updates are:

$$e'_w - \alpha_{\text{RCM}} \left(\sigma(f'(w)) - I_{[w \in R_{w_i}]}\right) \cdot e'_{w_i}$$

$$e'_{w_i} - \alpha_{\text{RCM}} \sum_{w} \left(\sigma(f'(w)) - I_{[w \in R_{w_i}]}\right) \cdot e'_w,$$

where $f'(w) = {e'_w}^\top e'_{w_i}$. We use two learning rates: α_{cbow} and α_{RCM}.

2.4 Parameter Estimation

All three models (cbow, RCM and joint) use the same training scheme based on Mikolov et al. (2013). There are several choices to make in parameter estimation; we present the best performing choices used in our results.

We use noise contrastive estimation (NCE) (Mnih and Teh, 2012), which approximately maximizes the log probability of the softmax objective (Eq. 2). For each objective (cbow or RCM), we sample 15 words as negative samples for each training instance according to their frequencies in raw texts (i.e. training data of cbow). Suppose w has frequency $u(w)$, then the probability of sampling w is $p(w) \propto u(w)^{3/4}$.

We use distributed training, where shared embeddings are updated by each thread based on training data within the thread, i.e., asynchronous stochastic gradient ascent. For the joint model, we assign threads to the cbow or RCM objective with a balance of 12:1(i.e. C is approximately $\frac{1}{12}$). We allow the cbow threads to control convergence; training stops when these threads finish processing the data. We found this an effective method

for balancing the two objectives. We trained each cbow objective using a single pass over the data set (except for those in Section 4.1), which we empirically verified was sufficient to ensure stable performances on semantic tasks.

Model pre-training is critical in deep learning (Bengio et al., 2007; Erhan et al., 2010). We evaluate two strategies: random initialization, and pre-training the embeddings. For pre-training, we first learn using cbow with a random initialization. The resulting trained model is then used to initialize the RCM model. This enables the RCM model to benefit from the unlabeled data, but refine the embeddings constrained by the given relations.

Finally, we consider a final model for training embeddings that uses a specific training regime. While the joint model balances between fitting the text and learning relations, modeling the text at the expense of the relations may negatively impact the final embeddings for tasks that use the embeddings outside of the context of word2vec. Therefore, we use the embeddings from a trained joint model to pre-train an RCM model. We call this setting Joint→RCM.

3 Evaluation

For training cbow we use the New York Times (NYT) 1994-97 subset from Gigaword v5.0 (Parker et al., 2011). We select 1,000 paragraphs each for dev and test data from the December 2010 portion of the NYT. Sentences are tokenized using OpenNLP[1], yielding 518,103,942 tokens for training, 42,953 tokens for dev and 41,344 for test.

We consider two resources for training the RCM term: the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) and WordNet (Fellbaum, 1999). For each semantic pair extracted from these resources, we add a relation to the RCM objective. Since we use both resources for evaluation, we divide each into train, dev and test.

PPDB is an automatically extracted dataset containing tens of millions of paraphrase pairs, including words and phrases. We used the "lexical" version of PPDB (no phrases) and filtered to include pairs that contained words found in the 200,000 most frequent words in the NYT corpus, which ensures each word in the relations had support in the text corpus. Next, we removed duplicate pairs: if <A,B> occurred in PPDB, we removed relations of <B,A>. PPDB is organized

---

PPDB		Relations	WordNet	Relations
Train	XL	115,041	Train	68,372
	XXL	587,439	(not used in	
	XXXL	2,647,105	this work)	
Dev		1,582	Dev	1,500
Test		1,583	Test	1,500

Table 1: Sizes of semantic resources datasets.

into 6 parts, ranging from S (small) to XXXL. Division into these sets is based on an automatically derived accuracy metric. Since S contains the most accurate paraphrases, we used these for evaluation. We divided S into a dev set (1582 pairs) and test set (1583 pairs). Training was based on one of the other sets minus relations from S.

We created similar splits using WordNet, extracting synonyms using the 100,000 most frequent NYT words. We divide the vocabulary into three sets: the most frequent 10,000 words, words with ranks between 10,001-30,000 and 30,001-100,000. We sample 500 words from each set to construct a dev and test set. For each word we sample one synonym to form a pair. The remaining words and their synonyms are used for training. However we did not use the training data because it is too small to affect the results. Table 1 summarizes the datasets.

4 Experiments

The goal of our experiments is to demonstrate the value of learning semantic embeddings with information from semantic resources. In each setting, we will compare the word2vec baseline embedding trained with cbow against RCM alone, the joint model and Joint→RCM. We consider three evaluation tasks: language modeling, measuring semantic similarity, and predicting human judgements on semantic relatedness. In all of our experiments, we conducted model development and tuned model parameters (C, α_{cbow}, α_{RCM}, PPDB dataset, etc.) on development data, and evaluate the best performing model on test data. The models are notated as follows: word2vec for the baseline objective (cbow or skip-gram), RCM-r/p and Joint-r/p for random and pre-trained initializations of the RCM and Joint objectives, and Joint→RCM for pre-training RCM with Joint embeddings. Unless otherwise notes, we train using PPDB XXL. We initially created WordNet training data, but found it too small to affect results. Therefore, we include only RCM results *trained* on PPDB, but show evaluations on both PPDB and WordNet.

Model	NCE	HS
word2vec (cbow)	8.75	6.90
RCM-p	8.55	7.07
Joint-r ($\alpha_{\text{RCM}} = 1 \times 10^{-2}$)	8.33	6.87
Joint-r ($\alpha_{\text{RCM}} = 1 \times 10^{-3}$)	**8.20**	**6.75**
Joint→RCM	8.40	6.92

Table 2: LM evaluation on held out NYT data.

We trained 200-dimensional embeddings and used output embeddings for measuring similarity. During the training of cbow objectives we remove all words with frequencies less than 5, which is the default setting of word2vec.

4.1 Language Modeling

Word2vec is fundamentally a language model, which allows us to compute standard evaluation metrics on a held out dataset. After obtaining trained embeddings from any of our objectives, we use the embeddings in the word2vec model to measure perplexity of the test set. Measuring perplexity means computing the exact probability of each word, which requires summation over all words in the vocabulary in the denominator of the softmax. Therefore, we also trained the language models with hierarchical classification (Mikolov et al., 2013) strategy (HS). The averaged perplexities are reported on the NYT test set.

While word2vec and joint are trained as language models, RCM is not. In fact, RCM does not even observe all the words that appear in the training set, so it makes little sense to use the RCM embeddings directly for language modeling. Therefore, in order to make fair comparison, for every set of trained embeddings, we fix them as input embedding for word2vec, then learn the remaining input embeddings (words not in the relations) and all the output embeddings using cbow. Since this involves running cbow on NYT data for 2 iterations (one iteration for word2vec-training/pre-training/joint-modeling and the other for tuning the language model), we use Joint-r (random initialization) for a fair comparison.

Table 2 shows the results for language modeling on test data. All of our proposed models improve over the baseline in terms of perplexity when NCE is used for training LMs. When HS is used, the perplexities are greatly improved. However in this situation only the joint models improve the results; and Joint→RCM performs similar to the baseline, although it is not designed for language modeling. We include the optimal α_{RCM}

in the table; while set $\alpha_{\text{cbow}} = 0.025$ (the default setting of word2vec). Even when our goal is to strictly model the raw text corpus, we obtain improvements by injecting semantic information into the objective. RCM can effectively shift learning to obtain more informative embeddings.

4.2 Measuring Semantic Similarity

Our next task is to find semantically related words using the embeddings, evaluating on relations from PPDB and WordNet. For each of the word pairs in the evaluation set <A,B>, we use the cosine distance between the embeddings to score A with a candidate word B'. We use a large sample of candidate words (10k, 30k or 100k) and rank all candidate words for pairs where B appears in the candidates. We then measure the rank of the correct B to compute mean reciprocal rank (MRR). Our goal is to use word A to select word B as the closest matching word from the large set of candidates. Using this strategy, we evaluate the embeddings from all of our objectives and measure which embedding most accurately selected the true correct word.

Table 3 shows MRR results for both PPDB and WordNet dev and test datasets for all models. All of our methods improve over the baselines in nearly every test set result. In nearly every case, Joint→RCM obtained the largest improvements. Clearly, our embeddings are much more effective at capturing semantic similarity.

4.3 Human Judgements

Our final evaluation is to predict human judgements of semantic relatedness. We have pairs of words from PPDB scored by annotators on a scale of 1 to 5 for quality of similarity. Our data are the judgements used by Ganitkevitch et al. (2013), which we filtered to include only those pairs for which we learned embeddings, yielding 868 pairs.

We assign a score using the dot product between the output embeddings of each word in the pair, then order all 868 pairs according to this score. Using the human judgements, we compute the swapped pairs rate: the ratio between the number of swapped pairs and the number of all pairs. For pair p scored y_p by the embeddings and judged $\hat{y}_p$ by an annotator, the swapped pair rate is:

$$\frac{\sum_{p_1,p_2 \in D} \text{I}[(y_{p_1} - y_{p_2})(\hat{y}_{p_2} - \hat{y}_{p_1}) < 0]}{\sum_{p_1,p_2 \in D} \text{I}[y_{p_1} \neq y_{p_2}]} \quad (4)$$

where $\text{I}[x]$ is 1 when x is true.

Model	PPDB						WordNet					
	Dev			Test			Dev			Test		
	10k	30k	100k	10k	30k	100k	10k	30k	100k	10k	30k	100k
word2vec (cbow)	49.68	39.26	29.15	49.31	42.53	30.28	10.24	8.64	5.14	10.04	7.90	4.97
word2vec (skip-gram)	48.70	37.14	26.20	-	-	-	8.61	8.10	4.62	-	-	-
RCM-r	55.03	42.52	26.05	-	-	-	13.33	9.05	5.29	-	-	-
RCM-p	61.79	53.83	40.95	65.42	55.82	41.20	15.25	**12.13**	7.46	14.13	**11.23**	7.39
Joint-r	59.91	50.87	36.81	-	-	-	15.73	11.36	7.14	13.97	10.51	7.44
Joint-p	59.75	50.93	37.73	64.30	53.27	38.97	15.61	11.20	6.96	-	-	-
Joint→RCM	**64.22**	**54.99**	**41.34**	**68.20**	**57.87**	**42.64**	**16.81**	11.67	**7.55**	**16.16**	11.21	**7.56**

Table 3: MRR for semantic similarity on PPDB and WordNet dev and test data. Higher is better. All RCM objectives are trained with PPDB XXL. To preserve test data integrity, only the best performing setting of each model is evaluated on the test data.

Model	Swapped Pairs Rate
word2vec (cbow)	17.81
RCM-p	16.66
Joint-r	16.85
Joint-p	16.96
Joint→RCM	**16.62**

Table 4: Results for ranking the quality of PPDB pairs as compared to human judgements.

Model	Relations	PPDB Dev		
		10k	30k	100k
RCM-r	XL	24.02	15.26	9.55
RCM-p	XL	54.97	45.35	32.95
RCM-r	XXL	55.03	42.52	26.05
RCM-p	XXL	**61.79**	**53.83**	**40.95**
RCM-r	XXXL	51.00	44.61	28.42
RCM-p	XXXL	53.01	46.35	34.19

Table 5: MRR on PPDB dev data for training on an increasing number of relations.

Table 4 shows that all of our models obtain reductions in error as compared to the baseline (cbow), with Joint→RCM obtaining the largest reduction. This suggests that our embeddings are better suited for semantic tasks, in this case judged by human annotations.

Model	α_{RCM}	PPDB Dev		
		10k	30k	100k
Joint-p	1×10^{-1}	47.17	36.74	24.50
	5×10^{-2}	54.31	44.52	33.07
	1×10^{-2}	**59.75**	**50.93**	**37.73**
	1×10^{-3}	57.00	46.84	34.45

Table 6: Effect of learning rate α_{RCM} on MRR for the RCM objective in Joint models.

4.4 Analysis

We conclude our experiments with an analysis of modeling choices. First, pre-training RCM models gives significant improvements in both measuring semantic similarity and capturing human judgements (compare "p" vs. "r" results.) Second, the number of relations used for RCM training is an important factor. Table 5 shows the effect on dev data of using various numbers of relations. While we see improvements from XL to XXL (5 times as many relations), we get worse results on XXXL, likely because this set contains the lowest quality relations in PPDB. Finally, Table 6 shows different learning rates α_{RCM} for the RCM objective.

The baseline word2vec and the joint model have nearly the same averaged running times (2,577s and 2,644s respectively), since they have same number of threads for the CBOW objective and the joint model uses additional threads for the RCM objective. The RCM models are trained with single thread for 100 epochs. When trained on the PPDB-XXL data, it spends 2,931s on average.

5 Conclusion

We have presented a new learning objective for neural language models that incorporates prior knowledge contained in resources to improve learned word embeddings. We demonstrated that the Relation Constrained Model can lead to better semantic embeddings by incorporating resources like PPDB, leading to better language modeling, semantic similarity metrics, and predicting human semantic judgements. Our implementation is based on the word2vec package and we made it available for general use [2].

We believe that our techniques have implications beyond those considered in this work. We plan to explore the embeddings suitability for other semantics tasks, including the use of resources with both typed and scored relations. Additionally, we see opportunities for jointly learning embeddings across many tasks with many resources, and plan to extend our model accordingly.

[2]https://github.com/Gorov/JointRCM

References

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. 2007. Greedy layer-wise training of deep networks. In *Neural Information Processing Systems (NIPS)*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning (ICML)*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research (JMLR)*, 11:625–660.

Christiane Fellbaum. 1999. *WordNet*. Wiley Online Library.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Association for Computational Linguistics (ACL)*, pages 873–882.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Conference on Natural Language Learning (CoNLL)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *International Conference on Machine Learning (ICML)*.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition. Technical report, Linguistic Data Consortium.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *Association for Computational Linguistics (ACL)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Association for Computational Linguistics (ACL)*.

Benjamin Van Durme and Ashwin Lall. 2010. Online generation of locality sensitive hash signatures. In *Association for Computational Linguistics (ACL)*, pages 231–235.