Explicit Representation of Antonymy in Language Modeling

Geoffrey Zweig
Microsoft Research
Redmond, WA USA
gzweig@microsoft.com

MSR-TR-2014-52

Abstract

The word vectors learned by continuous space language models are known to have the property that the vectors of synonymous words have cosine similarities close to one. To date, however, the relation of antonymy has not been captured in these models. In this paper, we demonstrate that by incorporating prior information from a thesaurus as an extra term in the language model objective function, the induced word vectors are useful not only for language modeling, but also for modeling both synonymy and antonymy. The learned vectors have the property that the vectors of antonymous words tend to have cosine similarities close to negative one, while synonymous words retain similarity close to positive one. We show that with a small penalty in language model perplexity, the induced word vectors do very well on a standard GRE test of opposites.

1 Introduction

Continuous space word representations have recently proven useful across a spectrum of tasks as diverse as language modeling (Bengio et al., 2003; Bengio et al., 2006; Schwenk and Gauvain, 2002; Mikolov et al., 2010; Mikolov et al., 2011b; Mikolov et al., 2011a); word tagging (Collobert and Weston, 2008; Turian et al., 2009; Yao et al., 2013; Mesnil et al., 2013); sentiment analysis (Socher et al., 2011; Socher et al., 2012); and machine translation (Auli et al., 2013; Kalchbrenner and Blunsom, 2013; Son et al., 2012; Schwenk et al., 2006). A variety of mechanisms have been proposed for generating such representations. Primary among them are Latent Semantic Analysis (Deerwester et al., 1990), continuous space language models (Bengio et al., 2003; Mikolov et al., 2010; Mnih and Teh, 2012), methods based on predicting words in a local window (Mikolov et al., 2013a; Mikolov et al., 2013b), multi-task learning (Collobert and Weston, 2008), and word-word co-occurrence models (Lebret et al., 2013).

In addition to being useful for classical tasks such as language modeling, continuous space word representations implicitly represent important semantic generalizations. It has long been observed that in these models, similar words tend to have similar representations; for example, "minister" and "official" will tend to have a high cosine similarity. More recently, it was shown that the representations learned by these models capture a much wider variety of syntactic and semantic phenomena (Mikolov et al., 2013c). Specifically, pairs of words that share a syntactic or semantic relationship tend to share the same vector offset with respect to each other. For example, the vector connecting "faster" to "fastest" will be almost parallel to the vector connecting "bigger" to "biggest."

Interestingly, when a continuous space language model is used, these properties occur in the learned word representations almost by accident, without any explicit support in the model. While this has the advantage that no information other than general text training data is necessary, it also has the disadvantage that certain relations like antonymy are difficult to learn. Previous approaches cannot distinguish between words like "hot" and "cold" because they are distributionally similar, and the models are based purely on distributional similarity. In this paper, we add *prior knowledge* into a language model, which allows us to model relations that cannot be learned otherwise - while jointly representing distributional information. In contrast to the implicit learning of relations in previous models, the main contribution of this work is to *explicitly* support semantic relations that would not otherwise be learned.

2 Related Work

Recent work in continuous space word representations has focused on efficient learning methods (Mikolov et al., 2013a; Mnih and Teh, 2012), and the degree to which deep or predictive models are necessary (Lebret et al., 2013). Testing has been focused on semantic and syntactic test sets from Microsoft Research and Google (Mikolov et al., 2013c; Mikolov et al., 2013a; Zweig and Burges, 2011). In (Mikolov et al., 2013a), it was shown that a formal language model is not necessary to learn vectors with interesting properties. More recently, (Lebret et al., 2013) show that good representations can be learned with a fairly simple co-occurrence model. This past work has used the *vector offset method* introduced in (Mikolov et al., 2013c) to answer questions. This method relies on the previously mentioned property that the unit vectors of all pairs of words which share a particular relationship exist at a constant vector offset with respect to one another. However, since antonymy is a reflexive property, the vector offset method, which is not reflexive, is sub-optimal to model it [1]. In contrast to prior work, we focus here on a reflexive relation.

While continuous space word representations have been the subject of much research, and have been used to improve performance on semantic tasks, e.g. (Collobert and Weston, 2008; Turian et al., 2009; Yao et al., 2013; Mesnil et al., 2013), there has been relatively little work in learning antonyms. The work of (Yih et al., 2012) is closest to ours. In this, the information in a thesaurus is used to determine word embeddings such that antonyms can be detected, via *Polarity Inducing Latent Semantic Analysis* (PILSA). However, the PILSA model can only be used to detect synonyms and antonyms. This key difference of this paper is that the prior knowledge in a thesaurus is used to augment a model based on distributional similarity; the resulting model is a classical language model, which additionally knows opposites. In comparison to the multi-task learning of (Collobert and Weston, 2008), we use a classical language model that produces a distribution over successor-words, and focus on the reflexive relation of antonymy.

[1] This is discussed further in Section 5

3 Model

The model we use is the log-bilinear language model (LBLM) due to (Mnih and Hinton, 2007), with the improvements in (Mnih and Teh, 2012). The main difficulty of using such a model is that the output distribution must be normalized over all the words in the vocabulary, which is prohibitively expensive for normal vocabulary sizes. One way of reducing the computational complexity is with a hierarchical output layer (Mnih and Hinton, 2009; Le et al., 2011; Mikolov et al., 2011b; Zweig and Makarychev, 2013), and another is with the recently developed method of Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012). In this paper, we adopt the latter due to its simplicity. We also extend the log-bilinear model by adding Maximum Entropy features; this has proven useful with recurrent neural networks (Mikolov et al., 2011a), and we observe a similar gain with the LBLM. Our choice of the LBLM is motivated by its compromise between simplicity, computational efficiency, and good performance. We now describe the model in detail.

3.1 Log Bilinear Model

The LBLM is an n-gram model in which an $n-1$ word history is used to predict a distribution over successor words. Two vectors are maintained for each word w: one w^h which is used when the word is in the history, and one w^p when it is being predicted. A separate diagonal scaling matrix A_i is associated with each position in the history, $i \in \{-1 \ldots -n+1\}$. The probability of a word y given the history $h = w_{-1}, w_{-2} \ldots w_{-n+1}$ is given by:

$$p(y|h) = \frac{\exp(\left(\sum_{i=-1}^{-n+1} A_i w_i^h\right) \cdot y^p)}{\sum_x \exp(\left(\sum_{i=-1}^{-n+1} A_i w_i^h\right) \cdot x^p)}$$

In its original formulation (Mnih and Hinton, 2007), training was done using stochastic gradient ascent so as to maximize the training data likelihood. This is computationally expensive due to the normalization in the denominator, which carries over into the gradient computation. We avoid this by training with NCE, which results in an approximately normalized model, without the need for explicit normalization. Due to space limitations, we refer the reader to (Mnih and Teh, 2012; Gutmann and Hyvärinen, 2010) for details.

3.2 Maximum Entropy Features

In previous work, maximum entropy features have been found to be an effective addition to a continuous space language model (Mikolov et al., 2011a). We incorporate these features into the LBLM using a hash table. The hash table notionally contains a real valued entry for every possible n-gram suffix. Denote the hash table by $T[]$, and a hash function mapping a word sequence into an integer between 0 and the hash-table size as hash(). The (unnormalized) probability of a word y given a history $h = w_{-1}, w_{-2} \ldots w_{-n+1}$ is then

$$P(y|h) \propto \exp\left(\left(\sum_{i=-1}^{-n+1} A_i w_i^h\right) \cdot y^p + T[\text{hash}(y)] + \sum_{i=-1}^{-n+1} T[\text{hash}(w_i \ldots w_{-1} y)]\right)$$

The reader will note that the use of a hash table means that due to collisions, an entry will not in general correspond to just one n-gram suffix. Except for very large datasets, this has not proved to be a problem, and the issue is fully discussed in (Xu et al., 2011).

3.3 Modeling Antonymy

As in (Yih et al., 2012), we use a thesaurus as a source of prior knowledge with which to model antonymy. However, rather than doing latent semantic analysis, which results in vectors that are unsuitable for language modeling, we add a term to the LBLM objective function that explicitly models synonymy and antonymy.

We define the set of words related to a word w by either synonymy or antonymy as related(w). We define a function polarity(x, y) that returns 1 when x and y are synonyms and -1 when they are antonyms. Further, let sim(x, y) be the cosine similarity between vectors x and y:

$$\text{sim}(x, y) = \frac{x \cdot y}{||x||\ ||y||}$$

We then define a semantic objective function of the model parameters Θ as:

$$\sum_{w} \sum_{x \in \text{related}(w)} \text{sim}(x, w)\text{polarity}(x, w) \qquad (1)$$

This function achieves its maximum when the cosine similarity between every pair of synonyms is 1 and the similarity between every pair of antonyms is -1.

Data	Train	Test	Vocab
Treebank	930k	82k	10k
Wikipedia	58.8M	234k	130k

Table 1: Number of words in language modeling data sources.

Model	Treebank	Wikipedia
LBLM	164	221
LBLM + ME	143	182
Kneser-Ney	161	200

Table 2: Perplexity of trigram models.

Equation 1 is a function defined on the word representations much like the L2 norm that is sometimes used in regularization; it is not a function of any development or test data like data likelihood. Thus, adding it to the objective function can be thought of as semantic regularization. As with other forms of regularization, we weight this by some value γ and add it to the overall objective function. We do not use any other form of regularization.

In preliminary experiments, we found that the input-side or predictive word vectors learned semantic information somewhat better than the output representations. Thus, the semantic regularization was applied to the input representations. This is consistent with past work using recurrent neural network models (Mikolov et al., 2013c).

3.4 Training

The model is trained with NCE and stochastic gradient ascent. We used Adagrad (Duchi et al., 2011), and parameters were updated after processing 40 sentences. For NCE, we used the unigram distribution raised to the $3/4$ power as the noise distribution (Mikolov et al., 2013b), and created 10 contrastive examples for each n-gram in the data. The contrastive samples were computed on-demand, so different samples were used in different passes through the data. The scaling matrices were initialized to the identity matrix, and the word representations to random unit vectors. Unless otherwise mentioned, these parameter settings, trigram models, and 100 dimensional word representations are used in the experiments.

The semantic objective function was optimized in mini-batches at each parameter update. As processing proceeds, we accumulate a list of all the words whose representations will be updated.

Word	LBLM-Similar Words	LBLM-Opposite Words	SemReg-Similar Words	SemReg-Opposite Words
pleasant	happy, colder, chilly, damp, lush	ensigns, vl, airstrike, taffy, launcher	likable, agreeable, personable, inoffensive, congenial	nasty, unpleasant, disagreeable, unlikable, displeasing
astute	resourceful, skillful, tactless, assertive, versatile	mascara, maffei, sarnath, caliban, forst	shrewd, sagacious, expedient, canny, insightful	unsubtle, tactless, crass, indiscreet, thoughtless
president	president-elect, vice-president, then-president, then-governor, secretary-general	stage's, accusatorial, 50-foot, blade's, cutler's	leader, chancellor, commander, director, boss	laggard, straggler, underling, vassal, junior

Table 3: Comparison of most-similar and most-opposite words, using representations from a regular LBLM and a Semantically-Regularized LBLM.

Then, at the end of a mini-batch, we compute the gradient of the semantic objective function with respect to the parameters of these words, and add a term to the overall gradient, before the update.

The per-iteration training time of a 3-gram, 100 dimensional LBLM for our 59M word training set was about 80 minutes on a 2.5GHz Xeon processor running single threaded. Adding the semantic constraints is computationally expensive because the objective function requires many pairwise comparisons, and vector normalization when computing cosine similarity. The runtime with semantic constraints increased to about 250 minutes per iteration. Convergence is typically achieved in under 20 iterations in both cases.

4 Experimental Results

4.1 Data

We present results for two datasets. For comparison with previously published results, we use the Penn Treebank dataset, *Linguistic Data Consortium* catalog number LDC1999T42, normalized as in (Mikolov et al., 2010). For larger-scale our experiments, we used a randomly selected 59M word subset of a 2010 Wikipedia snapshot. Following (Yih et al., 2012), we used the Encarta Thesaurus developed by Bloomsbury Publishing Plc[2] as the source of our prior knowledge. Table 1 shows the sizes of these data sets. The Bloomsbury Thesaurus contained about 50k words and 3M distinct synonymy/antonymy relations. While NCE training produces approximately normalized models, for test-set perplexity computations, we did exact normalization. The unnormalized perplexities resulting from NCE training were typically within 5 to 10% of the normalized perplexities.

To test the learned vectors' capacity to model antonymy, we used the GRE test set of (Mohammed et al., 2008). This consists of a set of multiple-choice questions which have the goal of identifying the word most opposite to a given word. Each question has five options, so guessing achieves 20%. We tuned model parameters on the development set of 162 questions, and report test results on the test set of 950 questions. We report results in terms of F1 score as in (Yih et al., 2012).

4.2 Basic Language Model

Table 2 illustrates the test-set perplexity for the LBLM with and without maximum entropy features. For the Treebank results, we used 50 contrastive samples per word in training, while for the Wikipedia set we used 10 to limit the computational expense. In both cases, the maximum entropy hash table was set to 100M entries. As can be seen, the LBLM augmented with Maximum Entropy features outperforms the standard discrete n-gram model with state-of-the-art Kneser Ney smoothing (Kneser and Ney, 1995).

4.3 Antonymy

To achieve a qualitative sense of model performance, Table 3 shows the most similar and most opposite words, using representations learned by both a regular LBLM and a semantically-regularized LBLM. It is immediately obvious that the regular language model does not represent antonyms at all. Moreover, many of the simi-

[2] http://www.bloomsbury.com/

Measure	LM	LM + ME	LM + SR	LM + SR + ME
Perplexity	221	182	305	209
GRE (F1)	0.121	0.136	0.742	0.741

Table 4: Perplexity and GRE score for models with and without semantic regularization.

lar words are actually opposites. For example, in a regular language model, "tactless" appears as very similar to "astute," and "damp" is similar to "pleasant." Interestingly, the semantically regularized model does a good job finding opposites even for words that have no formal antonyms. For example, the least similar words to "president" include "underling" and "vassal." Neither of these is explicitly listed as the opposite of "president" in the thesaurus, which indicates that the model has learned to effectively generalize.

In Table 4, we illustrate both perplexity and performance on the GRE test set. We see that by explicitly modeling antonymy in the objective function, we are able to learn word vectors that not only predictive for language modeling, but model antonymy as well. For comparison, if we discard the language modeling term in the objective function and simply optimize the semantic part, we achieve a performance of 0.771 in F1 measure. These are in line with the 0.70 reported in (Mohammed et al., 2008) and the 0.77 and 0.80 reported in (Yih et al., 2012) for Polarity Inducing Latent Semantic Analysis (PILSA) and PILSA followed by neural net training. Interestingly, the vectors learned by a regular language model actually do worse than chance (20%); we hypothesize that this is because the antonyms used in the test are distributionally very similar to the focus word.

We have systematically studied the sensitivity of the model to the hyper-parameters: the initial value of alpha in AdaGrad; the mini-batch size; the size of the maximum entropy hash table; the number of contrastive samples used in NCE; the dimensionality of the representations; the regularization weight; and the n-gram level. The model is generally insensitive to variations around the settings we have used, and space precludes tables of all the results. However, the effect of the regularization weight is interesting, and we present it in Table 5, which shows the tradeoff between perplexity and the ability to identify antonyms.

Measure	0.001	0.01	0.1
Perplexity	192	209	240
GRE (F1)	0.728	0.741	0.750

Table 5: Effect of semantic regularization weight.

5 Comparison with Vector Offset Method

Previous work has found that certain semantic and syntactic relationships can be identified in word embeddings through the vector offset method (Mikolov et al., 2013c). Consistent with the results of (Mnih and Teh, 2012), we have found that the LBLM does a good job in learning vectors in which the vector offset method can be useful. However, after training with antonym constraints, we found that the vector offset method performs poorly on tests of syntactic similarity (Mikolov et al., 2013c) and (Mikolov et al., 2013a). In future work we plan to explore methods of maintaining the vector offset property where it does well, while augmenting it in other cases such as antonymy.

It is also worth noting that the vector offset method can produce reasonable results for the "opposites" section of (Mikolov et al., 2013a); however, this set is restricted to pairs in which the negated form of a word (e.g. "illogical") always follows the non-negated form (e.g. "logical"). If we reverse the order of presentation, accuracy drops by a factor of 4, from 21% to 5%; yet the relation of antonymy still exists between the input words. In contrast, the representations learned in our proposed model capture antonymy in a completely reflexive way.

6 Conclusion

In this paper, we have developed a method for incorporating prior knowledge from a thesaurus into the word vectors learned by a language model. This may be thought of as a form of *semantic regularization* and allows us to learn continuous space word representations that encode semantic relations that are otherwise difficult to learn. We find that the word embeddings produced by this method give competitive performance on a standard GRE antonymy test, while at the same time serving as the basis for a state-of-the-art language model.

References

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *EMNLP*.

Y. Bengio, R. Ducharme, Vincent, P., and C. Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Reseach*, 3(6).

Y. Bengio, H. Schwenk, J.S. Senécal, F. Morin, and J.L. Gauvain. 2006. Neural probabilistic language models. *Innovations in Machine Learning*, pages 137–186.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(96).

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), Seattle, USA. Association for Computational Linguistics*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Hai-Son Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon. 2011. Structured output layer neural network language model. In *Proceedings of ICASSP 2011*.

R. Lebret, J. Legrand, and R. Collobert. 2013. Is deep learning really necessary for word embeddings? In *NIPS Deep Learning WOrkshop*.

G. Mesnil, He X., L. Deng, and Y. Bengio. 2013. Investigation of recurrentneural- network architectures and learning methods for language understanding. In *Interspeech*.

Tomas Mikolov, Martin Karafiat, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech 2010*.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. 2011a. Strategies for Training Large Scale Neural Network Language Models. In *Proceedings of ASRU 2011*.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2011b. Extensions of recurrent neural network based language model. In *Proceedings of ICASSP 2011*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv: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 *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, pages 641–648.

A. Mnih and G.E. Hinton. 2009. A scalable hierarchical distributed language model. *Advances in neural information processing systems*, 21:1081–1088.

A. Mnih and Y. Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *ICML*.

Saif Mohammed, Bonnie Dorr, and Graeme Hirst. 2008. Computing word pair antonymy. In *Empirical Methods in Natural Language Processing (EMNLP)*.

H. Schwenk and J.L. Gauvain. 2002. Connectionist language modeling for large vocabulary continuous speech recognition. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–765. IEEE.

H. Schwenk, D. Dchelotte, and J.L. Gauvain. 2006. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 723–730. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

L.H. Son, A. Allauzen, and F. Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–48. Association for Computational Linguistics.

J. Turian, L. Ratinov, Y. Bengio, and D. Roth. 2009. A preliminary evaluation of word representations for named-entity recognition. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*.

Puyang Xu, Sanjeev Khudanpur, and Asela Gunawardana. 2011. Randomized maximum entropy language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 226–230. IEEE.

K. Yao, G. Zweig, M-Y. Hwang, Y. Shi, and D. Yu. 2013. Recurrent neural networks for language understanding. In *Interspeech*.

Wen-tau Yih, Geoffrey Zweig, and John C Platt. 2012. Polarity inducing latent semantic analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1212–1222. Association for Computational Linguistics.

Geoffrey Zweig and Christopher J.C. Burges. 2011. The Microsoft Research sentence completion challenge. Technical Report MSR-TR-2011-129, Microsoft.

Geoffrey Zweig and Konstantin Makarychev. 2013. Speed regularization and optimality in word classing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8237–8241. IEEE.