Improving word embeddings to capture semantics

Harshal Godhia
harshals@seas

Derry Wijaya
derry@seas

Chris Callison-Burch
ccb@seas

Abstract

Most current models of semantic word representation exploit the distributional hypothesis: the idea that words occurring in similar contexts have similar meanings (Harris). Such representations (or embeddings) can reflect human intuitions about similarity and relatedness. For example, the word pairs *young-old*, *young-new* have similar context and occur together. It is easy to point out that relatedness is not the same as similarity, the second pair is more similar than the first. Through this work we wish to train embeddings so that we can specialize them to capture similarity rather than relatedness. To this end we test the embeddings generated by our model on WordSimilarity and Simlex dataset. Another significant contribution of the paper is to explore and provide a neural model baseline for re-ranking the PPDB corpus.

1 Introduction

Simlex-999 (Hill et al., 2014) task provides a way of measuring how well models capture similarity, rather than relatedness or association. This is hard because most language-based representation-learning models infer connections between concepts from their co-occurrence in corpora, and co-occurrence primarily reflects relatedness not similarity. Simlex999 is in contrast to the WordSim353 task (Finkelstein et al., 2001) which captures both relatedness and similarity. For example, for the pair (coast,shore) SimLex999 provides a score of 9 and so does WordSim353 but for the pair (clothes, closet) SimLex999 provides a score of 1.96 as compared to WordSim353 which scores it at 8. Our baseline model on the SimLex999 task is that of pre-trained Glove (Pennington et al., 2014) embeddings. They are trained from co-occurrence matrices and currently the most popular text embeddings used for downstream tasks. Starting with pre-trained glove embeddings our approach is to use the word pairs of PPDB to push similar words closer in the embedding space and dis-similar words apart.

Our second work of re-ranking PPDB is an experimental study where we consider the goal of re-ranking PPDB pairs through a neural model that is trained on human annotated pairs of paraphrases or a pretrained generic sentence encoder (which may have been trained in unsupervised fashion on large corpora like (Kiros et al., 2015) or on pre-text tasks like textual entailment (Bowman et al., 2015). Currently, PPDB provides scores based on a supervised model which is trained on 176 features engineered for this particular task. Given the unprecedented success of neural models we wish to explore the use of 1) standard sentence encoders trained on very large datasets transferred to the given task to decide paraphrase equivalences and 2) explore if a model can be trained from scratch. We train on a considerably small dataset of only 22,000 pairs of sentences. The baseline we consider is the word2vec model which gives a co-relation score of 0.4633 and the current state of art model is a supervised model with about 176 features giving a co-relation score of 0.7130. We explore this avenue since using a neural model decreases the dependence on feature engineering and the model we develop to score paraphrases can also be applied to other tasks like textual entailment, sentence similarity etc.

This paper is organized into sections which first introduce the data, the model followed by results on the test set. It is important to mention that all our results are based on a reduced vocabulary comprising of the words contained in PPDB.

This results in a significant out of vocabulary rate (OOV) which we intend to keep for future work. In contrast to word specialization techniques like (Faruqui et al., 2014) which are a post-processing approach. Our approach essentially entails fine-tuning the neural model.

2 Related Work

We consider our work to be motivated by that of (Wieting et al., 2015) and following its paradigm we train a skipgram model (Mikolov et al., 2013) on a semantic paraphrase lexicon PPDB(Ganitkevitch et al., 2013).Another avenue of related work is that of (Tissier et al., 2017) where the authors utilize dictionary word definitions to specialize word embeddings for tasks of semantic similarity and compare the performance with other models which learn from lexicons such as wordnet, ppdb etc. The high level concept of using external lexicons is not new. Some approaches require fine tuning where we re-train the model on the lexicon data while others like (Faruqui et al., 2014) are a post processing task and hence do not depend on the underlying model which was used to generate embeddings. There has been many works such as ours which aim to specialize word embeddings making them suitable for particular downstream tasks or datasets. Another avenue of work which has been studied is to enable these word embeddings to caputre word polysemy such as (Iacobacci et al., 2015). Most similar to our goal is the work of (Mrksic et al., 2016) which has the same motivation as ours: to separate out antonyms and bring together synonyms so as to increase the word similarity, although they take the post-processing approach called ”cross-fitting”. They further demonstrate the importance of this task in dialogue state tracking problems, where it is important to differentiate between for example, a pricey or an inexpensive restaurant.Another work which looks for symmetric patterns like X and Y so synonyms words can come closer in space.
To develop the neural model we compare word order preserving models, as studied in (Conneau et al., 2017) and un-ordered bag of words models as studied in (Iyyer et al., 2015). Qualitatively we choose to base our model on the un-ordered variant as the paraphrases which we work with have a maximum length of 6 with the average much smaller, indicating that LSTM and sequence models may be too complex. Further, (Iyyer et al., 2015) demonstrates that deep neural bag of words approach provide very competitive results. Recently released work by (Gardner et al., 2017) have also shown the effectiveness and efficiency of models not learning word orders necessarily. Another piece of related work which specializes word embedding based on context is (Melamud et al., 2016) which generated both a representation for the target word and for the context. Specializing word embeddings for specific tasks like sentiment analysis is common, this work (Tang et al., 2014) attempts to improve classification of tweets.

3 Data

3.1 Fine tune word embedding task

- Our input data is PPDB 2.0 of lexical kind of size L. We ignore records with a PPDB 2.0 score below 3.4 Further we ignore pairs where the edit distance between the two words is less than half of the minimum of the two word lengths

- When passing the data to the neural model we mini-batch. This is an important hyper-parameter of the model because with a very small size we have no learning. We tried different mini-batch sizes including 50, 64, 100, 128, 512, 1000 and found 100 to be optimal

3.2 PPDB re-ranking task

- For the re-ranking task the data comprises of 26556 phrase pairs which we reformat as 2286 phrase, list of paraphrases. These phrases are collected from a crowd sourcing task where annotators are given the task to assign a score between 1 and 5 for the phrase, paraphrase. We take the average of 5 annotators to be the score of similarity between the phrase and paraphrase. An important point to mention is that when building the test and train split one exclusively works on the dictionary format of phrase, list of paraphrases so that the model doesn’t end up memorizing information but can generalize easily to other kind of data and

4 Model

4.1 Fine tune word embedding task

Since we only want to specialize the embeddings and fine tune them. We adopt the skip gram model of word2vec and train the model on the PPDB. For a very crisp explanation of the word2vec skipgram model we cite (Goldberg and Levy, 2014). We adopt the log likelihood skipgram model which uses negative sampling for providing incorrect examples for every correct target word and context pair. In our work, each paraphrase pair of PPDB forms a correct target word and context pair. The number of negative examples is a hyper parameter of the model which we analyze in detail. The size of negative samples is chosen by validating model performance on the test set by varying choices 5, 10, 20 and 40. We observed that with a very large negative sample size the learning is unstable because it is not very discriminative, we found best test accuracy for negative sample size of 5. We study and experiment with the following techniques for negative sampling.

- *Random:* All the negative samples are chosen uniformly at random from the entire vocabulary. This approach has the advantage that by the process of randomization the words separate from a different set of negative words in each iteration resulting in a smoother manifold. But it is also possible that some of the random words chosen are synonyms or related with the target world

- *Bottomk:* For each word in the vocabulary we generate the 50 farthest or dissimilar words based on the initial word embeddings (glove). This guarantees that no word which is similar or related will be chosen as a negative sample. But the issue is that the same negative words are chosen in each epoch and thus the manifold generated is not smooth. Further, the bottom 50 words are often ill formed words or words which very rarely occur. Ideally we would prefer to have negative words to be well formed and meaning bearing words carrying almost the opposite semantic information

- *Topk:* For each word in the vocabulary we generate the 50 closest or similar words based on initial word embeddings (glove). It provides a guarantee that the words would be meaning bearing but run the risk of moving similar words apart by treating them as a negative example. Although some papers have shown this to work in our experiments this did not perform well

- *Random + Antonym + Topk:*This is the approach adopted in our work. We first select antonyms present of the target word in a lexicon (which is human annotated and generated on MTurk) and then randomly sample the remaining examples from the entire vocabulary ensuring that none of them are among the top 100 similar words. This combines the advantage of randomization which ensures a smooth manifold with the fact that no similar neighbors are moved apart

4.2 Re-ranking PPDB task

We now describe the neural model that we develop for the PPDB re-ranking task. The model follows a Siamese architecture (Mueller and Thyagarajan, 2016) where the phrase encoders are shared between both of the paraphrases and they pass through the encoder layer to generate phrase embeddings. We then compute the inner product, the absolute difference and the embeddings themselves pass through a series of feed forward fully connected networks ending in a MSE loss to make a prediction between 1-5 and compare against the average of the annotator scores.

5 Implementation

5.1 Fine tuning word embedding task

For the skip gram neural model we experimented with the following combinations of optimizers. Regardless to say, this parameter is very crucial to the models performance. For each combination we evaluate model performance on the test set to determine the best value

- SGD with learning rate 0.3. This has the problem that towards epoch 20 the loss plateaus out and we do not see any progress in learning. This plateauing of learning is not resolved even when we adaptively decrease the learning rate by a factor (considered 3 to 10)

- To avoid the issue of manually having to adapt the learning rate we switched to Adagrad with an initial learning rate of 0.005. Although it may seem too small a learning rate to start with we notice that the loss decrease is monotonic until it starts to converge around epoch 60. This also gives us the smallest loss value and best accuracy

5.2 Re-ranking PPDB task

- Each feed forward block comprises of a linear layer, followed by batch normalization followed by ReLU layer
- The batch size is set to 100 and we use the Adagrad optimizer with a learning rate of 0.0001
- To simplify the model instead of having an embedding layer inside the network, in the pre-processing itself for each training point we compute the average phrase embedding

6 Results

6.1 Fine tuning word embedding task

Model A trained on data after ignoring pairs with PPDB2.0 score of less than 3.4. Model B trained on data after ignoring pairs with PPDB2.0score of less than 3.1. Scores in [] indicate the vector embeddings for words not in vocabulary were filled by default glove embeddings and SimLex999 scores were computed in table 1

6.2 Re-ranking PPDB task

When we consider the entire 26000 pairs of phrases and split it into 20000 training pairs of phrases and 6000 test phrases we get a spearman co-relation of 0.758 on the test phrases. But here we run the problem of defining the test phrases incorrectly since it is likely that of the phrases which occur in the test set they may have paraphrases in the training set. So the model could end up memorizing some of the results and thus overfit, leading to a weak performance on unseen phrases. So instead we define our dataset as phrase, paraphrases. This considerably reduces the data size to 2286 such entries. We chose 2000 of these entries as training points and the remaining 286 as test points. Using this modified data split and format we obtain a spearman co-relation of 0.22. This is considerably lower than the current state of art of 0.71 but does better than the random baseline, interestingly it performs lower than word2vec as well. There are a couple of insights that we have from this experiment

- We use a model with a capacity of 580100 parameters or approximately 0.5 M parameters, this will in all likelihood over fit the data as we have only 2000 data points of phrase, paraphrases

7 Qualitative Analysis

Consider the word pair (sleep, dream). Pre-trained glove reports a similarity score of 0.42 in comparison to (sleep, awake) which it assigns a score of 0.71. This shows that similar meaning words are given a score much slower than frequently co-occurring words. Our goal was to fix this problem, and to this extent our model predicts a score of 0.58 for (sleep, dream) and 0.77 for (sleep, awake). It is too much to expect the model to separate out the words (sleep, awake) but the model makes a fairly significant attempt at bringing (sleep, dream) together. An almost 38% improvement. Similarly, we notice for the word pairs (Hi,Bye) and (Hi,Hello) word2vec gives a difference of 7% as compared to Model B which gives a difference of 17% thereby confirming the hypothesis that our fine tuning model improves word similarity. We provide more such examples in table 2

8 Future work

As we noticed in this paper, one can fine tune and specialize embeddings for similarity instead of the default (relatedness/co-occurrence). We also noticed that the neural model baseline for re-ranking PPDB highly over-fits the small human annotated paraphrase data of 2000 phrase, paraphrases and hence in future work we explore the potential of training the model on large dataset such as (Bowman et al., 2015)

9 Appendix

Our code can be found at (Godhia)

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR* abs/1705.02364. http://arxiv.org/abs/1705.02364.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2014. Retrofitting word vectors to semantic lexicons. *CoRR* abs/1411.4166. http://arxiv.org/abs/1411.4166.

Table 1: Results on Simlex, and word similarity

	WS-rel	WS-sim	SimLex999	Vocab size (out of 999)
Pre-trained Glove[+fill OOV]	0.46[0.46]	0.61[0.57]	0.29[0.26]	484[999]
PPDB-trained	0.35	0.56	0.34	484
+ Controlled sampling *A[+fill OOV]	0.40[0.41]	0.57[0.546]	0.426[0.32]	484[999]
+ Controlled sampling *B	0.47	0.543	0.378	635

Table 2: Improving similarity scores

Pair	Pre-trained	Mode B
Young, New vs Young, Old	0.58 vs 0.70	0.63 vs 0.69
Hi, Bye vs Hi, Hello	0.32 vs 0.39	0.49 vs 0.66
Happy, Sad vs Happy, Joy	0.68 vs 0.62	0.76 vs 0.77
Take, give	0.93	0.90

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*. ACM, New York, NY, USA, WWW '01, pages 406–414. https://doi.org/10.1145/371920.372094.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-burch. 2013. Ppdb: The paraphrase database. In *In HLT-NAACL 2013*.

Andrew Gardner, Jinko Kanno, Christian A. Duncan, and Rastko R. Selmic. 2017. Classifying unordered feature sets with convolutional deep averaging networks. *CoRR* abs/1709.03019. http://arxiv.org/abs/1709.03019.

Harshal Godhia. ???? `https://github.com/hsgodhia/independent-study`.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR* abs/1402.3722. http://arxiv.org/abs/1402.3722.

Zellig Harris. 1954. Distributional structure. *Word* 10(23):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR* abs/1408.3456. http://arxiv.org/abs/1408.3456.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 95–105. http://www.aclweb.org/anthology/P15-1010.

Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daum III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL (1)*. The Association for Computer Linguistics, pages 1681–1691. http://dblp.uni-trier.de/db/conf/acl/acl2015-1.htmlIyyerMBD15.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 3294–3302. http://dl.acm.org/citation.cfm?id=2969442.2969607.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *CoNLL*.

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

Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Counter-fitting word vectors to linguistic constraints. *CoRR* abs/1603.00892. http://arxiv.org/abs/1603.00892.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*.

Julien Tissier, Christophe Gravier, and Amaury Habrard. 2017. Dict2vec : Learning word embeddings using lexical dictionaries. In *EMNLP*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *CoRR* abs/1506.03487. http://arxiv.org/abs/1506.03487.