Exploring the Effects of External Semantic Data on Word Embeddings

Zihua Liu
Stanford University
Stanford, CA
zliu19@stanford.edu

William Hang
Stanford University
Stanford, CA
willhang@stanford.edu

Brian Zhang
Stanford University
Stanford, CA
bhz@stanford.edu

Abstract

Distributed word embeddings have been a groundbreaking development that is widely used in many deep Natural Language Processing tasks such as Machine Translation, Question Answering and etc. However, despite its success, currently popular word embedding methods such as GloVe, Skipgram or CBOW only consider distributional statistics of words on their own without any external semantic information. To address this concern, we explore the hypernym-hyponyms relations scraped form WordNet as an external source of semantics in order to build hierarchical semantics into the word embeddings. In this paper, we will present some of our approaches, results tested on various standard benchmark tests, and a brief analysis of these test results.

1 Introduction

Distributed word embeddings are semantic vector space models of language that aims to represent each word as a low dimension vector by leveraging neural networks and the context of the words. Among many methods to acquire such embeddings, one method, Global Vectors or GloVe [1], stood out and has gained much popularity because this method is able to take advantage of both the family of global matrix factorization methods and the family of local context window methods. With the success of GloVe, word embeddings have been used to various Natural Language Processing tasks such as question answering [2], machine comprehension [3], neural machine translation [4] and more.

However, there are intrinsic problems with the current training scheme that GloVe does not address. Firstly, current word embedding training methods revolve solely around distributional statistics of words such as co-occurrence statistics and window-based context. GloVe does not explicitly utilize the meanings contained in the words themselves throughout its training. There is a lot more granularity contained in the meaning of words that summary statistics simply cannot capture. For example, words with directly opposite meanings or descendants of the same category of nouns can be substituted quite freely within the context, making the difference hard to capture by distributional statistics although the meaning of the words themselves can be opposite or completely unrelated. Another problem of GloVe is that distributed representation of word vectors does not highlight relations among words, particularly hierarchy of words based on the abstractness of its meaning. As a result, we are motivated to explore the effects of external semantic information contained in WordNet. We extract 6989 pairs of hyponyms to hypernym data. An example of our extracted data is:

$$\texttt{Entity : \{abstraction, thing, physical\_entity\}}$$

In the following sections, we will present some methods we explored with these external semantics information extracted from hypernym-hyponym relations in WordNet and its test results on common evaluation tasks such as word similarity, categorization, and analogy. We will also provide a comprehensive analysis of these test results.

2 Related Work

There are past researches that attempts to leverage extra semantic information. For instance, Liu et al [5] proposed to use WordNet data to enhance Skip-gram performance based on ordinal knowledge constraints. They manually and carefully defined constraints between the similarity of words based on conceptual relations. For instance, they defined that similarity between words with same hypernym should be greater than similarity between words with different hypernyms. They updated their embeddings by jointly training skip-gram model and their knowledge constraint.

Another group of researchers, Faruqui et al [6], proposes a refinement of GloVe and Skip-gram models by encouraging semantically linked words from lexicon to have similar word embeddings. Specifically, they took a graphical approach and defined vocabulary list V as vertices and an *ontology* Ω that encodes the semantic relation as an undirected graph. They first train the embeddings independently and then retrofit the word vectors to minimize difference among linked words in the ontology.

Finally, Iacobacci et al [7] proposes another direction to improve word embedding. Iacobacci proposed to build distributed representation of each individual word sense to minimize ambiguity among means of words. They first take BabelNet, a combination of WordNet and other semantic lexicons to generate underlying word sense inventory. Then they generated an annotated text corpus from Wikipedia 2014. Ultimately, the trained a CBOW model to maximize log-likelihood of each word sense with respect to its context to get the final sense embedding.

3 Utilizing WordNet Dataset

As described earlier we choose to use WordNet data as our external semantics source. Specifically, for every word in WordNet, we extract all of is possible hyponyms and generate relations from the extracted hyponyms to the original word, the hypernym. Our reasoning behind choosing WordNet dataset and the hypernym-hyponyms relation is as follows:

First, hypernym-hyponyms relation puts emphasis on semantic hierarchy intrinsic to meanings of word. This hierarchy is organized by abstractness of the word and an inclusion relationship. In organization of WordNet, each word contains a list of other sub-categories as its hyponyms. This inclusion relation is what we mainly aim to highlight in our models.

Secondly, intra-hyponyms relation exposes semantic similarity among words. For the words that are hyponyms of a share hypernym, they are both sub-categories of the hypernym. It is clear that there are semantic similarities shared among these words. For instance, hyponyms of the word fruit should share the core semantic meaning of fruit, i.e.: an edible, tangible, usually sweet substance.

Lastly, WordNet data is abundant, structured, yet varied. WordNet contains 117,000 different synsets that cover most of common and important words that appear in the English language. Despite the size of the corpus, all of the synsets are linked together with only a small set of conceptual relations such as hypernym-hyponym relation we employ in this paper, synonym and antonym relation, etc.

4 Approach

We tested a multitude of methods by varying the model architecture, prediction task, loss function, and embedding initialization. In this section we detail the list of methods attempted, and why they were attempted. Empirical results are presented in Section 5.

4.1 Standard Recurrent LSTM

In this experiment, we built a standard recurrent LSTM to accept hyponym word embeddings as inputs. The word embeddings used were GloVe. One hyponym embedding would be accepted at each time step. At the last time step, the final hidden state would be used to either (1) predict the correct hypernym word embedding or (2) predict the correct hypernym in a sparse classification task.

The first task naturally lent itself to using L2 loss, and the second task lent itself to using CE loss. We tried both methods to observe how the final word embeddings varied with the prediction task. In both tasks, gradients were back-propagated into the word embeddings, and at the end of training, word embeddings were evaluated using similarity tests.

The reason both tasks are distinct is that the first task would allow for more generalization since loss wasnt decided by a classification task; L2 would allow the predicted hypernyms to differ from the ground truth hypernyms. The second task would optimize for the principle of compositionality that the LSTM would learn to combine the hyponyms to learn and generate higher level concepts.

The unrolled recurrent LSTM had 3 layers, and 10 time steps. We would retrieve the last hidden state and either project it into the embedding space of 300 for the embedding prediction task or the vocabulary space of 20,000 for the classification task.

4.2 Randomly initialized word embeddings

We decided to try initializing the word embeddings randomly instead of using pre-initialized GloVe embeddings to test if the model would generate better results by starting from scratch. The embeddings were 300 dimensional with values uniformly chosen from 0 to 1.

4.3 Feedforward model

Worried about the potential that using the LSTM would impose a nonexistent expectation that the list of hyponyms was temporal/time dependent, we tried using a feedforward neural network to accept an averaged vector of hyponym embeddings and perform the classification task described in Section 4.1. The reason we accept an averaged vector of embeddings is that the hypernyms had a differing number of available hyponyms, and we did not want to zero-pad or pad with an element in the hyponym set to skew the prediction.

The model can be described with:

$$\mathbf{h} = mean\left(\begin{bmatrix} hypo_1 \\ hypo_2 \\ ... \\ hypo_n \end{bmatrix} \mathbf{W}_{proj}\right)$$

$$\mathbf{h} = ReLU(\mathbf{h}\mathbf{W}_1 + \mathbf{b}_1)$$
$$\hat{y} = \mathbf{h}\mathbf{W}_2 + \mathbf{b}_2$$

Where $\mathbf{W}_{proj}$ projects hyponym embeddings from the embedding space to a higher dimensional projection space of 600, and the output dimension of $\mathbf{W}_1$ is 300 and the output dimension of $\mathbf{W}_2$ is the vocabulary size.

4.4 Attentional model

In order to observe if we could force the model to explicitly compose the result from the input hyponym embeddings, we trained an attentional model over the embeddings to produce the correct hypernym. This way, we could preserve the embeddings up until we took a weighted average vector and fed the vector through an MLP to generate embeddings.

Specifically, the model generated a context vector c through a 2 layer feedforward network by accepting the concatenated hyponym embeddings and producing a context vector of dimensionality at most that of the original embedding space. This is described by:

$$\mathbf{h} = [hypo_1; hypo_2; ...; hypo_n]\,\mathbf{W}_{c1} + \mathbf{b}_{c1}$$
$$\mathbf{c} = \mathbf{h}\mathbf{W}_{c2} + \mathbf{b}_{c2}$$

Where $hypo_i$ is the ith hyponym for the hypernym, and $\mathbf{W}_{c1}$ has an output dimension of 500, and $\mathbf{W}_{c2}$ has an output dimension of 300.

We would then take the dot product of each of the hyponym vectors in a training example and the concatenated vectors to generate a score distribution that reflected how similar each hyponym vector was to the context vector. We normalized this score distribution with softmax. This can be done by:

$$\mathbf{score} = softmax\left(\begin{bmatrix} hypo_1 \\ hypo_2 \\ \ldots \\ hypo_n \end{bmatrix} \mathbf{c}^\top\right)$$

Then, we generated a vector that was the weighted average vector of the hyponym vectors, with weights defined by the score distribution by:

$$\mathbf{w}_{avg} = \mathbf{score} \begin{bmatrix} hypo_1 \\ hypo_2 \\ \ldots \\ hypo_n \end{bmatrix}$$

To obtain this weighted average, we left multiply the hyponym matrix with the score distribution. Finally, the weighted average vector was fed through an MLP to generate the final predicted hypernym. In this method, we only optimized for L2 loss because we wanted the model to learn compositionality while not constraining itself to satisfy a prediction task; our hope was that this would encourage the vectors to diverge from GloVe.

4.5 Co-occurrence matrix modifying

The GloVe model uses a large, sparse co-occurrence matrix that encodes how many times a given word is seen in the context of another word. except with the addition for each (hypernym, hyponym) pair, if we observe the hyponym in a given context, we also add a small amount to the cooccurrence count for the hypernym. In particular, if word w_1 is observed in the context of word w_2, we not only add 1 to the cooccurrence counts for the pair (w_1, w_2), but we also add some tunable constant $c < 1$ (through some trial and error, we found $c = 0.01$ to be a reasonable value, so this is used in the Experiments section) to the cooccurrence counts of (h, w_2) for every hypernym h of w_1. Intuitively, this should help when the training corpus is small; in this case it should allow the corpus to be augmented with useful semantic data.

5 Experiments

To test the word embeddings that resulted from our various models, we used the common metrics of word similarity, word categorization purity, and analogy. These are among the tests suggested in [8]. We trained 300-dimensional word embeddings using various models and various corpus sizes. The 6B word corpus is the original corpus used in [1] to train GloVe; we simply used the publicly available resulting pre-trained vectors as an initialization. The 17M word corpus is text8, publicly available at [9]. The 1M word corpus is the first million tokens of text8.

For word similarity, we used the Wordsim353 dataset [10], which consists of 353 pairs of words labelled with word similarity scores on a scale of 1 to 10, ignoring any pair which contained a word that did not appear in the vocabulary. The score on this test is given by the Spearman r coefficient between the labelled scores and the cosine similarity between the embeddings of the two words being tested.

For word categorization, we used the concrete noun and verb categorization dataset provided for ESSLLI 2008 [11], which consists of 44 nouns and 45 verbs, which allow for five clustering tasks: 6-, 3-, and 2-way noun clustering; and 9- and 5-way verb clustering. We ran a standard K-means clustering algorithm using our embeddings and again only the words that are in our vocabulary. We evaluated the clusterings on *purity*, which is described e.g. in [12]:

$$\text{purity} = \frac{1}{N} \sum_k \max_j |w_k \cap c_j|$$

	Size	Sim	Cat	Ana
GloVe (pretrained)[1]	6B	60.1	**61.5**	85.0
LSTM[1,2]	- -	**60.6**	61.0	84.2
Feed-forward[1,2]	- -	59.5	61.2	**85.5**
Attention network[1,2]	- -	56.5	59.7	81.9
GloVe	17M	51.9	48.7	**35.0**
+ hypernym co-occurrence	17M	**55.1**	**52.4**	34.4
GloVe	1M	41.4	47.1	5.0
+ hypernym co-occurrence	1M	**49.5**	**48.3**	**5.1**

Figure 1: *Data from runs. Scores are in percentage; higher is better. Results are only comparable using the same training corpus.*
[1]*Restricted to those words that appeared in our hypernym training set*
[2]*Initialized with and then (after training) concatenated with pretrained GloVe vectors, so that the vectors are an augmentation of the GloVe vectors rather than a replacement for them*

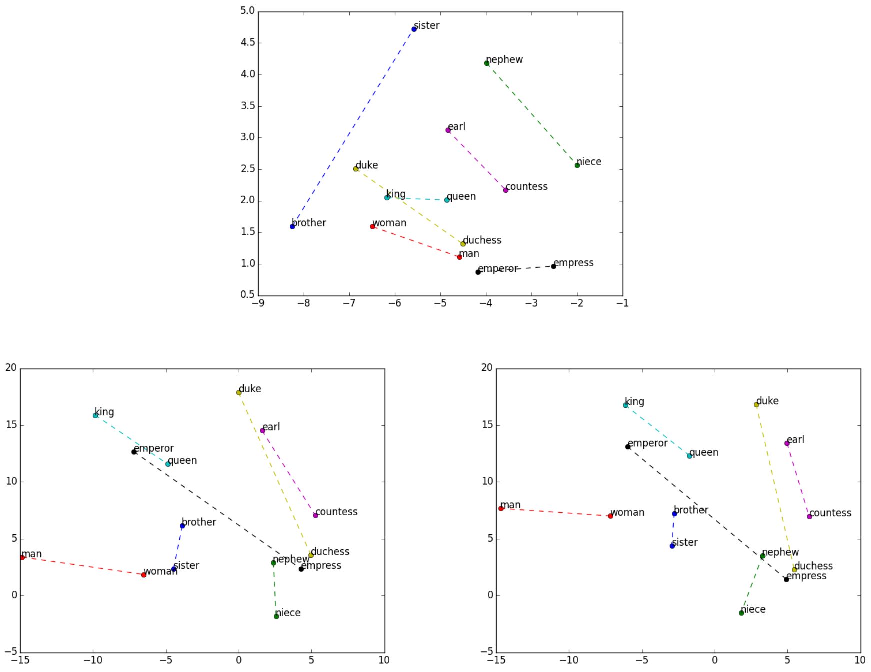

Figure 2: *Visualizations of the word vectors generated by various methods, using PCA.*
Top: *Attention network, initialized with pretrained GloVe vectors, before concatenation*
Bottom-Left: *GloVe vectors trained on text8 (17m)*
Bottom-Right: *GloVe vectors trained on text8 with hypernym co-occurrence counts*

where w_k denotes the set of words that K-means outputs to be in cluster k, c_j denotes the set of words in cluster j, and N is the number of words. Since K-means is not guaranteed to converge to local optima, we run the clustering test 100 times on each of the five tasks (each with random initialization), and report the overall average purity. It should be noted that placing every word in the same cluster attains a purity of 34.9%, so we should expect scores much higher than this.

For analogy, we use the evaluation dataset and code provided with the GloVe source code [1], which consists of 19544 analogy examples: 7416 semantic and 10411 syntactic. The test metric is simply the proportion of examples ($w_1 : w_2 :: w_3 : w_4$) on which w_4 is the embedding nearest $w_2 - w_1 + w_3$.

In Fig. 1 we see that augmenting the hypernym counts was by far the most successful idea, outperforming GloVe in multiple categories in the 1M and 17M corpora. (due to resource constraints, we were unable to train GloVe with hypernym co-occurrence counts on the full 6B word corpus).

The other methods were less successful. A likely reason for this is that the other methods put too much emphasis on one structure of a word (namely its place in the hypernym-hyponym graph) and therefore loses other structures that contribute to the meaning of a word. Augmenting hypernyms is less "intrusive" a method, in that the GloVe training process remains unchanged; the only changes are made to the co-occurrence matrix. This allows us to keep the advantages of GloVe while still incorporating the WordNet data, yielding a performance improvement.

In Fig. 2, it becomes more clear why training to predict hypernyms was not a successful technique: while the resultant vectors can predict hypernyms very well, it was easy for the vectors to lose other semantic meaning: for example, in the top image in the figure, we see that, for example, $king - queen$ points to the left, but $man - woman$ points to the right, opposite to what we expect; and $brother - sister$ has completely lost its direction. Therefore the vectors have lost some information by training to predict hypernyms.

Conversely, we notice that whether or not we add the hypernym co-occurrence counts has, as expected, a rather minimal effect on vectors, only shifting them slightly. (the effect will be most likely more pronounced on rarer words, but none of the words chosen for the visualization appear to be rare enough for this to be prominent) This means that, as noticed above, adding co-occurrence counts keeps the advantages and most of the nuance of the GloVe model, *while* adding more useful data, so it is unsurprising that the resultant vectors would perform better.

6 Conclusion

We tried several ideas to augment word embeddings using semantic information, in particular hypernym-hyponym relations, to augment word embeddings. By far the most successful was directly manipulating the co-occurrence counts, a technique which allowed us to outperform GloVe on small corpora such as the text8 corpus. We believe that this general technique of augmenting corpora using semantic data such as hyponym-hypernym relations can be used to improve word vector performance when the vectors are trained on small corpora.

In further experimentation, we will attempt new models and training objectives:

1. Building an autoencoder to jointly predict the hypernym and reconstruct a concatenated vector of hyponyms. We think this will allow our model to learn both a prediction task and learn a low dimensional latent representation of the hyponyms, which may allow the model to generalize better.
2. Adding tree or graph distance into the loss function for GloVe, where we not only attempt to minimize the original loss with context and focal words as parameters, but also attempt to predict the distance between the context and focal words.
3. Leveraging a larger or more diverse corpus of data to attempt to learn compositionality and hierarchy. Hyponym-hypernym relations comprise a relatively small and limited dataset. Expanding this to include (for example) synonyms and antonyms may also help look at the

7 Acknowledgements

We give acknowledgements to Arun Chaganty and Richard Socher for being our project mentors, and to Arun for suggesting modifying the GloVe co-occurrences to incorporate some semantic information and retraining.

References

[1] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[2] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," *CoRR, abs/1506.07285*, 2015.

[3] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603*, 2016.

[4] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[5] Q. Liu, H. Jiang, S. Wei, Z.-H. Ling, and Y. Hu, "Learning semantic word embeddings based on ordinal knowledge constraints.," in *ACL (1)*, pp. 1501–1511, 2015.

[6] 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.

[7] I. Iacobacci, M. T. Pilehvar, and R. Navigli, "Sensembed: Learning sense embeddings for word and relational similarity.," in *ACL (1)*, pp. 95–105, 2015.

[8] M. Baroni, G. Dinu, and G. Kruszewski, "Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proceedings of Association for Computational Linguistics (ACL)*, vol. 1, 2014.

[9] M. Mahoney, "About the test data." `http://www.mattmahoney.net/dc/textdata.html`, 2011.

[10] 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*, pp. 406–414, ACM, 2001.

[11] "Shared tasks from the esslli 2008 workshop." `http://wordspace.collocations.de/doku.php/data:esslli2008:start`.

[12] C. D. Manning, P. Raghavan, H. Schütze, *et al.*, *Introduction to information retrieval*, vol. 1. Cambridge university press Cambridge, 2008.