Learning Hypernymy over Word Embeddings

Neha Nayak
nayakne@stanford.edu

Abstract

Word embeddings have shown promise in a range of NLP tasks; however, it is currently difficult to accurately encode categorical lexical relations in these vector spaces. We consider one such important relation – hypernymy – and investigate the feasibility of learning a function in vector space to capture it. We argue that hypernymy is significantly harder to capture than the analogy tasks word embeddings are traditionally evaluated on. We show that a simple neural network outperforms previous systems at *classifying* hypernymy, and present experiments for learning a function to *predict* the hypernym of a word in a vector space.

1 Introduction

Word embeddings such as GloVe [1] and Word2Vec [2] have shown promise in a variety of NLP tasks. These word representations are constructed to minimise the distance between words with similar contexts. According to the distributional similarity hypothesis, this means that similar words should have similar representations, however they make no guarantees about more fine-grained semantic properties.

It has been shown that continuous word embeddings encode various simple lexical relations, such as *singular-plural* or *country-capital* as offsets in vector space. However, the completeness of these embeddings for semantic tasks has not yet been fully established. Vilnis et al. [3] has shown that word embeddings that encode a word as a distribution in space, rather than a point, outperform other distributed representations on tasks that require knowledge of the uncertainty of a concept. Recently, Faruqui et al. [4] has shown that retrofitting word embeddings to encode information from semantic lexicons such as Wordnet[5] results in a significant improvement in semantic tasks. Further, Socher et al. [6] showed that incorporating WordNet hypernymy features into an already successful sentiment classifier improves end-to-end F1.

Hypernymy, or the *is-a* relation, is an important lexical relation for NLP tasks. However, the extent to which word embeddings encode hypernymy has not been explored as extensively as the relations mentioned above. Knowledge of the hypernymy relation is critical for tasks such as Question Answering [7], Natural Language Inference [8], and Coreference Resolution [9].

It would be appealing if information about hypernymy were evident from the word representations themselves, removing the need for complementary resources. This would also assuage certain issues inherent to semantic lexicons. First, these lexicons are high-precision and thus are not always able to generalise to all terms in the vocabulary, whereas a soft measure of hypernymy based on the embeddings themselves would be generalise more easily. This would also allow a smoother distinction between related word senses. Finally, from a purely theoretical perspective, it is interesting to determine whether these embeddings are rich enough to capture hypernymy.

Most foregoing work on hypernymy has focussed on the task of *classifying* pairs of words according to whether or not the relation of hypernymy holds between them. We show that using word embeddings and a feedforward neural net classifier, it is possible to outperform existing work on the classification task.

Given the success of word embeddings at predicting simpler relations, such as *country-capital*, and the encouraging results of the classification task, we introduce a novel hypernymy *prediction* task, and explore the extent to which hypernyms can be predicted in vector space. This is a much harder, but also more useful task than classification. We show that we significantly outperform baselines on hypernymy prediction.

2 Task Description

2.1 Classification Task

In the classification task, a pair of words $\langle w_1, w_2 \rangle$ is given, and the objective is to classify the pair as a positive example if w_2 is a hypernym of w_1, or a negative example otherwise.

Classification of hypernyms has been carried out with some success using co-occurrence-based vectors. Weeds et al. [10] represented word pairs with the difference or concatenation of PPMI-weighted co-occurrence vectors [11], and reported the performance of various classifiers. Of the classifiers, a linear SVM achieved the best performance.[1]

Recently, [14] pointed out that using linear SVMs, as foregoing work has done, reduces the classification task to that of predicting whether in a pair of words, the second one has some general properties associated with being a hypernym. This is due to the fact that multiplicative interactions between the features of the hyponym and hypernym vector do not contribute to the classification decision. As such, it is to be understood that experiments using the difference of vectors are not exactly carrying out the task of classifying hypernymy as much as detecting whether the generality of two terms in a pair differ.

2.2 Prediction Task

An emergent property of word embeddings is that certain relations between words can be encoded as vector offsets between their representations [15]. Prior work in this area has focused primarily on word analogy tasks. Given a small set of word pairs that exemplify some function from tokens to tokens (such as "X is a city in state Y"), the task is to predict the image of some unseen word in this function ([15, 1]). This problem has also been explored extensively using traditional word vectors ([16, 17, 18]).

We define a novel prediction task, in which, given the representation $v_{hypo} \in \mathbb{R}^d$ of some word w_{hypo}, the objective is to predict $v_{hyper} \in \mathbb{R}^d$ which is the representation of its direct hypernym.

This task is more difficult than the analogy tasks for various reasons. First, we expect that accuracy on this task will be strongly affected by polysemy. A word with multiple senses is likely to have multiple, distinct tokens that can accurately be described as the direct hypernym for one or more of its senses. As such, the relation of hypernymy is not a function, and so modelling it as a function from $\mathbb{R}^d$ to $\mathbb{R}^d$ inherently incurs some amount of error. Second, hypernymy is a transitive relation. A word such as *cat* has multiple hypernyms – *feline*, *carnivore*, *placental*, *mammal* – and which of the hypernyms is the 'direct' hypernym is not well defined.

These theoretical concerns are depicted in the diagrams below. It is clear from observing even just three hypernyms and their hyponyms, in Figure 1, that a simple offset or affine transformation is unlikely to capture the relation of hypernymy for all words. Figure 2 indicates that the direct hypernym according to WordNet is not always the most cognitively salient hypernym. We note that in the hypernymy tree depicted in Figure 2, the links at a particular level in the tree (e.g. $\langle dog, canine \rangle$, $\langle cat, feline \rangle$, $\langle elephant, pachyderm \rangle$) are similar to each other and dissimilar to links at other levels in the tree. However, in the absence of a reliable measure of the generality of a particular vector, this property cannot be exploited.

Levy et al. [19] explained the reason why terms related by analogy are separated by a simple offset in vector space – words that participate in an analogy are alike in all dimensions of similarity except one. For example, a king and queen would share all contexts related to being a person, and being

[1] Roller et al. [12] and Baroni et al. [13] also find that an SVM classifier achieves the maximum classification accuracy

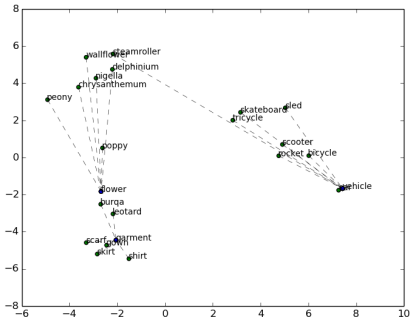

Figure 1: Three hypernyms and their hyponyms

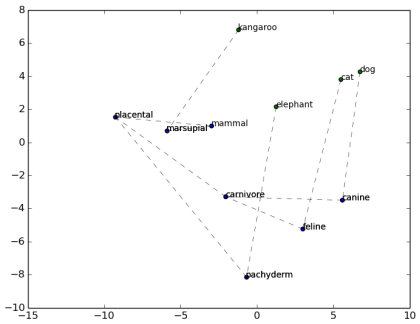

Figure 2: A subtree of the hypernymy tree

a royal, but would differ in the context related to being a man and a woman, respectively. In the operation $king - man + woman$, this amounts to reducing the probability of *man*-associated contexts and increasing the probability of *woman*-associated contexts represented by the resultant vector.

It is not immediately apparent, however, that the same arguments can be made for words in the hypernym-hyponym relation. This would require the existence of contexts that co-occur selectively with more general or less general words. However, since a word is more general or less general only in relation to another word. Levy et al. [14] found that, surprisingly, certain contexts do appear selectively with more general terms - for example, the word *such* appearing after a word indicates hypernymy, while *such* appearing before a word indicates hyponymy. This is reminiscent of the lexico-syntactic pattern *"x such as y"*, famously used in Hearst [20]. We attempt to leverage this observation to improve performance on prediction.

3 Classification Task

3.1 Data

The data for the classification task is taken from Weeds et al. [10]. Two datasets are used, BLESS, consisting of a subset of word pairs from the BLESS dataset[21], and WORDNET, consisting of a larger set of word pairs, constructed similarly, taking pairs from WordNet. Both datasets consist of pairs of words $\langle w_1, w_2 \rangle$, labelled such that in a positive example, w_2 is a hypernym of w_1, and in a negative example, w_2 is a word that is semantically related to w_1, but not a hypernym. This ensures that the classifier is in fact learning to detect the relation of hypernymy between words, as judgments based solely on distributional similarity would not be sufficient for accurate classification. In contrast to the prediction task, 'indirect' hypernyms (e.g. $\langle cat, entity \rangle$) are considered positive examples of hypernymy in this dataset.

3.2 Models

Following Weeds et al. [10], we represent each pair of words by the vector difference of their embeddings, using GloVe embeddings trained on Wikipedia. We then apply a feed-forward neural network to classify whether the pair is in the hypernymy relation.

	BLESS			WordNet		
	k-Fold	Dev	Test	k-Fold	Dev	Test
WEEDS	74.0	–	–	75.0	–	–
MAJORITY	50.0	50.6	52.3	50.0	51.5	50.3
1-LAYER	**76.4**	**90.4**	**93.0**	76.5	74.7	**74.3**
2-LAYER	74.4	88.3	89.8	**76.6**	**75.0**	72.7

Table 1: Performance of our neural network classifiers and prior work [10] on existing hypernymy classification datasets. We compare against prior work on their 10-fold cross-validation evaluation. We additionally compare the majority class baseline, a 1 layer neural network, and a two layer neural network on the development and test splits we generated.

The computation for the ith training example is described below, where $x^{(i)}_{hypo}$ and $x^{(i)}_{hyper}$ are the embeddings of the words in the ith pair, y is the output classification, j ranges from 1 to n in a n-layer neural network, and f is the $tanh$ nonlinearity:

$$\begin{aligned} z^{(i)}_0 &= x^{(i)}_{hyper} - x^{(i)}_{hypo} \\ z^{(i)}_j &= f(W_j z^{(i)}_{j-1} + b_j) \\ y^{(i)} &= softmax(z^{(i)}_n) \end{aligned} \tag{1}$$

Models were trained using SGD. L2 regularization and dropout were found not to improve performance. Hidden layers of size 500 were used. The input embedding size was cross validated over 100, 200 and 300 dimension embeddings.

3.3 Results

We report our results in Table 1. Weeds et al. [10] evaluated their models using 10-fold cross-validation, with the additional constraint of prohibiting shared words between the training and test set of each fold; note that this makes this evaluation more difficult. We additionally split the data into a standard train/development/test split, and report development and test numbers alongside the cross-validation results.

A 1-layer NN using GloVe embeddings outperforms existing systems on the BLESS and the more general WORDNET dataset. We observe that adding a second layer to the neural network deteriorates test performance, although Dev accuracy improves. Additional layers caused Dev error to deteriorate; these results are not reported. Comparable results from using either the concatenation or difference of the embeddings indicate that the difference of the vectors encapsulates most of the information relevant to the classification task.

4 Prediction Task

4.1 Data

For the prediction task, data was prepared using the word pairs in the hypernymy tree of WordNet. We use every pair of words $\langle w_1, w_2 \rangle$ such that w_2 is the direct hypernym of w_1 and both w_1 and w_2 are hyponyms of *entity* in WordNet. For each word pair, we use GloVe embeddings in 100 dimensions. GloVe vectors in 300 dimensions were also tested, however since these produced comparable results to the 100-dimension vectors, the results are not reported here. These vectors were selected due to their generality and as they have been shown to improve performance on other NLP tasks.[22, 23]

4.2 Models

In each model, we learn a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$, such that if $v_1 \in \mathbb{R}^d$ is the embedding of some word w_1, then $f(v_1) \in \mathbb{R}^d$ is the embedding of its direct hypernym. We denote the true embedding of the hypernym as $y^{\star(i)}$, and the predicted embedding of the hypernym as $y^{(i)}$.

Offset We implement a basic model, **Offset**, that mimics the operation described in [15]. The predicted vector in this model is obtained by adding a vector offset to the hyponym's representation. This model has a closed form solution given the training data, that minimises the distance of the predicted vector to the actual hypernym.

Mikolov et al. [15] uses an offset derived from a prototypical pair for the relation. In the absence of such an archetypal pair for hypernymy, we calculate the offset as the average offset between all pairs in the training set. This averaging method was observed to improve performance in Mikolov et al. [15].

$$\begin{aligned} v_{offset} &= \frac{1}{m}\sum_{i=1}^{m} y^{*(i)} - x_{hypo}^{(i)} \\ y^{(i)} &= x_{hypo}^{(i)} + v_{offset} \end{aligned} \tag{2}$$

Affine The simplest model we put forward is the **Affine** model, in which we learn an affine transformation from hyponym to hypernym. We use the mean Euclidean distance as the loss between $y^{\star(i)}$ and $y^{(i)}$, and learn the parameters W and b.

$$y^{(i)} = W x_{hypo}^{(i)} + b \tag{3}$$

FFNN Subsequently, we learn feedforward neural networks with increasing hidden layers. We use models of the following form, in which W_j and b_j are learned parameters, n varies between 3 and 6, and f is the $tanh$ nonlinearity.

$$\begin{aligned} a_1^{(i)} &= x_{hypo}^{(i)} \\ a_j^{(i)} &= f(z_j^{(i)})(j > 1) \\ z_j^{(i)} &= W_j a_{j-1}^{(i)} + b_j \\ y^{(i)} &= z_n^{(i)} \end{aligned} \tag{4}$$

These models are henceforth referred to as **FFNN1**, **FFNN2**, etc. For simplicity, we constrain the number of hidden units in each hidden layer to be equal. Hidden layer sizes were chosen to be scaled relative to the input size, where scaling factors of 0.5, 0.75, 1.0 and 1.5 were used. Models were trained using SGD with L2 regularisation. The objective function minimised was the mean Euclidean distance between $y^{\star(i)}$ and $y^{(i)}$.

Proto Levy et al. [14] indicated that word vectors encode some notion of generality, which allows classifiers to detect whether a particular word is likely to be a hypernym, regardless of the other word in the pair. To leverage this property, we construct an auxiliary vector that is intended to represent contexts of a prototypical hypernym. This was done in two ways - by taking the sum of all the hypernym vectors in the training set, and by taking the sum of offsets between all hyponym-hypernym pairs in the dataset. The latter method was more successful, and is reported on here. Note that in the second case, the prototypical hypernym vector is equal to the offset vector in **Offset**, and the **FFNN+Proto** is effectively interpolating between the **Offset** and **FFNN** models. In these models, the cost is a linear combination of the objective from the **FFNN** model and the Euclidean distance between the $y^{(i)}$ and the prototypical hypernym vector, where the relative weight of the two components is a hyperparameter.

NegSample Since the metric $precision@k$, described in Section 4.3 is not directly minimised by minimising the average cosine distance or Euclidean distance, we use negative sampling to minimise a more relevant quantity. The new loss function is defined as follows :

$$Loss(x_{hypo}; \theta) = 1 - cos(\hat{y}_{hypo}, y^{\star}_{hypo}) + \sum_{i=1}^{k} max\{0, cos(\hat{y}_{hypo}, x_{w_k}) + \textit{margin}\} \quad (5)$$

k is the number of negative samples – k words were selected by uniformly sampling from the vectors in the training set. The value of $margin$ used was 0.5.

4.3 Results

We report performance on our novel prediction task in Table 2. Following Mikolov et al. [2], we measure $precision@1$ and $precision@10$ – whether the predicted vector falls within the top-1, or top-10 nearest neighbors of the actual hypernym vector. We also report $precision@40$ – corresponding to the correct vector being in the top 0.01% of candidate vectors.[2]

	P@1		P@10		P@40		Test		
	Train	Dev	Train	Dev	Train	Dev	P@1	P@10	P@40
NN	0.38	0.47	1.99	1.80	5.12	4.76	0.40	1.96	5.19
Offset	0.69	0.84	6.96	6.63	14.84	14.21	1.07	7.55	14.55
Affine	1.48	1.37	17.24	14.29	34.08	30.72	1.37	15.43	32.22
FFNN1	2.18	1.94	19.87	16.39	37.42	32.66	-	-	-
FFNN2	2.31	2.17	19.94	16.58	37.49	33.57	-	-	-
FFNN3	2.40	2.21	20.09	16.73	37.96	34.18	-	-	-
FFNN4	**2.76**	**2.48**	**21.33**	**17.07**	**38.98**	**34.15**	**2.19**	**17.72**	**35.18**
FFNN1+proto	2.26	2.06	19.53	16.50	37.36	33.46	-	-	-
FFNN2+proto	2.20	2.13	19.80	16.12	37.68	32.39	-	-	-
FFNN3+proto	2.48	2.21	20.18	16.58	37.58	33.16	2.10	17.05	34.49

Table 2: Performance on the prediction task test set, in percent precision. NN is taking nearest neighbors; OFFSET is the offset method in [2]; AFFINE learns an affine transformation. FFNN are feedforward neural network models, +PROTO indicates that distance to a prototypical hypernym was included in the objective. Results are reported for 100-dimensional word embeddings.

An important thing to note here is the mismatch between training and test objectives: except in the negative sampling models, at training time, we are optimising cosine or Euclidean distance, whereas at test time we are finding k-best nearest neighbours.

The size of the output space is illustrated by the naïve random baseline – selecting a random vector from the vocabulary results in a $precision@k$ of 0% for all three values of k.

As a more realistic baseline, we use the **NN** model. In this model, for a word w_1 with embedding v_1, the predicted vector is the GloVe vector whose cosine distance to v_1 is minimal – its nearest neighbour. Since semantically related words are expected to be close in vector space, this should be a strong baseline. Any improvements above this baseline indicate that a model is learning a notion of hypernymy that is richer than semantic similarity. However, this baseline too does not exceed 0.5% $precision@k$, which is further evidence of the difficulty of the prediction task.

The **Offset** model was able to achieve $precision@1$ of 29.2% [15] for syntactic relations between nouns. Previous comparable explorations based on semantic relations use a small dataset annotated to allow computation of rank correlation, and, as such, report only rank correlation and not $precision@k$. In our experiments, **Offset** achieves $precision@1$ of 0.84% on the dev set. This is the best possible performance for this model, as there is a closed-form solution minimising loss on the training set.

[2]The size of the GloVe vocabulary is 400,000.

As expected, **Affine** outperforms **Offset**, and we observe increased precision upon the addition of hidden layers. Most notable, $precision@40$ shows an increase of 16% between **Offset** and **Affine** models, and thereafter continues to increase.

The additional consideration of the prototypical vector improved precision in the case of the **FFNN1** model. This may indicate that there are indeed contexts that are characteristic for hypernyms, however, this observation alone is not sufficient to account for the relation between hyponym and hypernym.

As the negative sampling model performed very poorly, the results are not reported.

Because of time constraints, models with over 4 hidden layers (for the **FFNN** model) and over 3 layers (for the model with the **FFNN+proto** model) were not tested, although dev precision had not begun to decrease.

A feed-forward neural net with four hidden layers achieved the best $precision@k$ on the dev set for all values of k. It was able to achieve a $precision@1$ of 2.19% on the test set, more than double the precision achieved by the offset model used in previous work on word relations.

Polysemy To determine the extent to which polysemy affected prediction accuracy, we carried out preliminary experiments restricting the word pairs to those in which both the hypernym and hyponym were monosemous. Although it was encouraging that experiments on these sets achieved higher precision, note that the Nearest Neighbours baseline also increases. This seemed to indicate that gains were solely due to monosemous words generally being lower-frequency and hence falling in a sparse area of the vector space. It was determined that accounting for polysemy would not be the most productive direction in which to continue work.

	P@1	P@10	P@40
NN	0.73	4.74	9.47
Offset	1.00	19.20	29.23
Affine	2.15	16.76	31.23

Table 3: Performance on the prediction task, restricting to hypernymy pairs where the hyponym is monosemous, using 300-dimension embeddings.

5 Conclusion

We have shown that commonly employed word embedding spaces encode hypernymy to a sufficient extent to outperform prior work on the hypernymy classification task. We have also demonstrated the potential of word embedding spaces in the task of predicting hypernyms, achieving $precision@1$ of up to 2.19%, and $precision@40$ of up to 35.18%. The models described take advantage of recent insights on the failings of existing work on hypernym classification, and show promising preliminary results for the hypernym prediction task.

Acknowledgements : Gabor Angeli (Stanford NLP group) provided extensive input for this project.

References

[1] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word representation. *EMNLP*, 2014.

[2] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, 2013.

[3] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*, 2014.

[4] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

[5] George A Miller. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[6] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP-CoNLL*, 2012.

[7] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building Watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79, 2010.

[8] Bill MacCartney. *Natural language inference*. PhD thesis, Stanford University, 2009.

[9] Vincent Ng and Claire Cardie. Improving machine learning approaches to coreference resolution. In *ACL*, Stroudsburg, PA, USA, 2002.

[10] Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. Learning to distinguish hypernyms and co-hyponyms. In *COLING*, 2014.

[11] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[12] Stephen Roller, Katrin Erk, and Gemma Boleda. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the Twenty Fifth International Conference on Computational Linguistics (COLING-14), Dublin, Ireland*, 2014.

[13] Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *EACL*, 2012.

[14] Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics â AS Human Language Technologies (NAACL HLT 2015), Denver, CO*, 2015.

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

[16] Peter D Turney. A uniform approach to analogies, synonyms, antonyms, and associations. In *Coling*, 2008.

[17] Peter D Turney. Distributional semantics beyond words: Supervised learning of analogy and paraphrase. *arXiv preprint arXiv:1310.5042*, 2013.

[18] Amaç Herdadelen and Marco Baroni. Bagpack: A general framework to represent semantic relations. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*. Association for Computational Linguistics, 2009.

[19] Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. Linguistic regularities in sparse and explicit word representations. *CoNLL-2014*, page 171, 2014.

[20] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

[21] Marco Baroni and Alessandro Lenci. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, 2011.

[22] Samuel R Bowman. Can recursive neural tensor networks learn logical reasoning? *arXiv preprint arXiv:1312.6192*, 2013.

[23] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.