Performance study of *Unsupervised Morphology Induction Using Word Embeddings* under different embedding models

Raja Gunasekaran
rgunasek@sfu.ca

Abstract

Soricut and Och [2015] proposed an unsupervised, language agnostic method to extract morphological rules and build a morphological analyzer. Their model improves on the state of art for word similarity in the morphologically-rich Stanford Rare-word dataset. For this project, we implemented the method proposed by Soricut and Och [2015] and studied the performance of their model under different word embeddings. To speed up computation, we used only top 100,000 words to extract the morphological rules. The pretrained word embeddings from Mikolov et al. [2013a,b] and Pennington et al. [2014] gave Spearman correlation of 22.9 and 22.5 respectively on RW dataset. By inducing morphological transformation and generating vector representation for rare and unknown words, we were able to improve the spearman correlation to 24.5 for SGNG and 25.9 for GloVe.

1 Introduction and Related works

Word embedding models like Bengio et al. [2003], Mikolov et al. [2013b], Pennington et al. [2014] etc., learn word representation in continuous vector space where similar words are expected to occur closer to each other. These models have been successfully used in many natural language processing tasks. The vector representation for word in most of these models primarily depend on word co-occurrence or context words within a window size capturing syntactic and semantic properties of natural languages.

One of the major challenges for many embedding models is their inability to handle out of vocabulary words. This problem is more pronounced in languages with rich morphology. A word in morphologically rich languages encodes more information (such as gender, number, tense) as compared to morphologically poor languages which rely on word order and context. Soricut and Och [2015] proposed a language-agnostic, heuristic method to capture morphological transformations by exploiting regularities present in word representations obtained using Skip-Gram model [Mikolov et al., 2013b]. Their method automatically induces morphological rules and transformations to represent them as vectors in the same embedding space. During testing, out of vocabulary words can be mapped into the same vector space using the learned morphological transformations. Their method achieved state-of-the-art performance on word similarity task using various standard datasets in 2015.

Since then, there have been other models that achieved state of art correlation on RW dataset. These results have been shown in Table 1. Bojanowski et al. [2017] incorporate subword information like character n-grams into the skipgram model. Sun et al. [2016] proposed a method to integrate both external contexts and internal morphemes to learn better word representation. Barkan [2017] proposed a bayesian neural word embedding model that relies on a Variational Bayes solution for the SG objective. Sanu et al. [2017] use a technique called Fixed-size ordinally forgetting encoding (FOFE) which has the ability to seize large contexts while discriminating contexts that are farther away as being less significant.

Embeddings	Method	Spearman Correlation on Stanford RW dataset
Soricut and Och [2015]	Skip Gram + Morphology	0.41
Bojanowski et al. [2017]	Subword Information Skip Gram	0.48
Sun et al. [2016]	Context + Morphology	0.52
Barkan [2017]	Bayesian Skip-Gram	0.50
Sanu et al. [2017]	FOFE+SVD	0.50

Table 1: State of art performance of different models on RW dataset

2 Approach

In this project, I studied the work of Soricut and Och [2015] by implementing their method and evaluating their algorithm under different pre-trained word embedding inputs from Mikolov et al. [2013a,b] and Soricut and Och [2015]. I am working on extending this study to different language and more embeddings. In this algorithm, the morphological rules are learned as follows.

1. Extract candidate rules like *('suffix', 'ies', 'y')* from vocabulary V and evaluate their quality in the pre-trained embedding space.
2. Generate morphological transformation from the above candidate rules and build a cyclic, multi-graph representing words as nodes and edges as morphological transformations.
3. Build a normalized acyclic graph with 1-to-1 morphological mapping from the above graph.
4. Map the rare/out of vocabulary words in the same vector space using morphological transformation using the graph.

3 Implementation Details

The implementation was done in python. I used gensim to work with word vectors and networkx graph library to build the graph. For find nearest neighbour in vector space, Annoy[1] was used. Major chunk of the time was spent on optimizing the run-time and memory usage of the program.

3.1 Candidate Rule Extraction

To speed up development process, pre-trained word embeddings from Mikolov et al. [2013b], Pennington et al. [2014] were used. For every pair (w_1, w_2) in vocabulary V, all possible prefix and suffix substitutions from w_1 to w_2 of predefined length 6 are extracted. For each rule r, its support set S_r - set of all word pair that obey rule r, is extracted.

$$S_r = \{(w_1, w_2) \in V^2 | w_1 \xrightarrow{r} w_2\}$$

To speed up computation, I split the vocabulary into separate chunks and ran them in separate cores. The candidate rule extraction over 1 million words from word2vec takes days to run on a 24 cores CPU machine. I couldn't fit all possible candidate rules for 3millions words in the memory. In the experiments below, I down-sampled the Vocabulary size to 100,000 most common words for rule extraction.

3.2 Evaluate candidate rules

The hit rate for the rules is computed using cosine rank. For every word-pair combination in $S_r \times S_r$, we calculate the percent of the time similairity rank is higher than the threshold t_{rank}. For word pair $((walking, walk), (talking, talk))$, the rank of $(walking, walk+ \uparrow d_{talk\_})$ was computed. In the Soricut and Och [2015] paper t_{rank} was 100. In the experiments, t_{rank} was set to 30 since the vocab size was smaller. It can be seen that meaning preserving rules like $suffix : ed : ing$ receive high hit rate while non meaning preserving rule receive low hit rate.

Since calculating, the exact nearest neighbour and their rank was costly, I used approximate nearest neighbour algorithm using k-d tree. First, all the word vectors are indexed using k-d tree using N trees. For this experiment, N was set to 100. Higher value of N will result in more accurate results but takes longer to calculate. Once this is done, finding the nearest neighbour takes $O(log(n))$ in the best case to $O(n)$ in the worst case.

[1] https://github.com/spotify/annoy

3.3 Generate Morphological Transformation

For each rule r, the best direction vector $\uparrow d_w$ is computed greedily. The direction vector $\uparrow d_w$ which explains most of the word pair in support is selected recursively for all unexplained word pairs. By doing this, we will have the morphological transformations for each rule.

These morphological transformations are subject to further evaluation using cosine similarity and cosine similarity rank. The cosine similarity threshold was set to t_{cosine} = 0.5 and cosine rank threshold was set to t_{rank} = 30. If this threshold is not met, the rules are removed.

The morphological transformations also have a graph based interpretations. A labelled, weighted, , directed multi-graph G^V_{Morph} is created from the transformations where nodes are the words in vocabulary and edges are the morphological transformation with $(rank, cosine)$ as the weights. From all the candidate morphological transformations 1-1 mappings are selected based on the the following conditions.

- edges are considered only if they start from higher count word and end at lower count word.
- if multiple edges exist, the ones with minimal rank $rank$ is selected.
- if multiple edges still exist, the one with maximum $cosine$ is selected.

This normalizes the graph to weighted directed graph D^V_{Morph} where low occurring words are mapped to high occurring words using a direction vector.

3.4 Mapping Rare and Unknown Words

The word vectors for out of vocabulary words, and low frequency words can generated by mapping them to nodes in D^V_{Morph} using the morphological rules. V_c is the set of all low frequency words and OOV words.

$$V_c = \{w \in V | C \leq count(w)\}$$

All the morphological rule $r \in R$ are applied on words $w \in V_c$ and if any of them results in a word $w' \in D^V_{Morph}$ then that word is mapped into the graph using the direction vector for rule r. This allows us to get meaning full representation for Rare and Unknown words at testing time.

4 Challenges

There were two major challenges in the implementation of the algorithm.

4.1 Rule Extraction

Extracting all candidate rules for all the words was very memory intensive and time consuming. The candidate rules for all 400,000 words in glove embeddings was 60GB in size which I was unable to fit in RAM to work with. To mitigate this, I split up the vocabulary into 10,000 chunks and for every word in each vocabulary chunk I compare against all the words in vocabulary and save the results in disk instead of holding it memory. Finally, I merge the results from all the 10,000 file using persistent storage. For improving the running time, I used a 24 core machine to run these chunks in parallel reducing the run time by factor of ~ 20.

4.2 Nearest Neighbour

Find the exact k-nearest neighbour can take $O(knd)$ time where n is number of points in vector space and d is dimension of each vector. With large vocabulary, getting the hit rate for each candidate rule takes days to run. I used an Approximate nearest neighbour algorithm from Annoy library which uses kd tree to index all the vector This sped up the computation by 500x. While, this may affect the quality of the morphological transformation slightly, this allowed for much faster experimentation.

5 Data

For the experiments in this projects, I used pretrained word embeddings from Mikolov et al. [2013a,b], Pennington et al. [2014] and Bojanowski et al. [2017]. In their work, the authors used words vectors of 500 dimensions. In this experiment, I used embeddings of size 300 dimensions for SGNG and 50 dimensions for Glove. For evaluation on English, Stanford English Rare-Word (RW) from Luong et al. [2013] was used. The RW dataset has 2034 word pairs with a higher degree of English morphology compared to other word-similarity datasets.

6 Experiments

For this project, I focused on re-implementing work of Soricut and Och [2015] and evaluating the spearman correlation ρ of the algorithm on a word similarity task. For this task, the word representations that capture the semantics of the word better should give higher correlation.

In the algorithm, the word representation in the vector space determines the quality of morphological rules. The authors use SGNS method (Mikolov et al. [2013a,b]) to create word embeddings. A more sophisticated model that is designed to capture the morphological relationship between words better should produce higher quality of morphological rules. This in turn should give higher correlation in the word similarity task. The correlation will be measured using word embedding generated from different methods like Pennington et al. [2014]. I was hoping to include the result of using Embeddings from Bojanowski et al. [2017], which capture sub-word information. But I was not able to get the results in time.

7 Results and Analysis

The spearman correlation ρ on RW dataset, for this implementation and other word embeddings models are shown in Table 1. The results for SkipGram (300 dimensions) and Glove (50 dimension) was obtained using publicly available word embeddings. This is the reason for the huge difference in correlation between models shows in Table 1 and Table 2. From Table 2, it can be seen that inducing morphological transformations improves the spearman correlation of the word pairs. By using this method, vector representations for $\sim$ 10% more words were obtained (1782 words using glove6B and 1996 words using glove6B + Morph). The 1%-2% increase in correlation indicate that vector representation for rare and unknown word(10%) were almost as good as vector representations for the known words. The authors report an increase of 6% in their results. There are couple of difference in the implementation from the Soricut and Och [2015] paper that may have hindered the algorithm to better morphological transformations.

For the project, the morphological rules were extracted only from top 100,000 words from the vocabulary out of millions of words. This was done to allow quick turn around time for implementation and experimentation of the algorithm. Generally, common words are morphologically poorer as compared to rarer words. Extracting morphological rules from all the words in vocabulary may result in further increase in the correlation.

Word Embeddings	**Spearman Correlation** $\rho \times 100$
SG	22.9
SG + Morph	24.5 **(+1.6)**
glove6B	22.5
glove6B + Morph	25.9 **(+2.4)**

Table 2: Performance of pre-trained and morphological transformation induced word embedding on Stanford RW Dataset

For the approximate nearest neighbour algorithm, I used 100 trees to build the indexing of all the vectors in the vector space. This allows for faster querying at the cost of precision. Building index using more number of tree will allow for higher precision at the cost of querying time.

8 Future Work

In my implementation, I was not able to get as much increase in correlation as reported in the paper. As a next step, I would like to run the experiments to get the performance gain reported in the paper. The following are some of the extensions that I am planning to do.

- Extract morphology from all words instead of extracting from only top 100,000 words. This required some code refactoring to optimize memory usage.
- Use nearest neighbour algorithm under high precision settings.
- Adding more morphologies like reduplication, multiple suffix/prefix, and infix morphology.
- Do the same experiments for languages with richer morphology than English like German, Turkish, etc..

References

Oren Barkan. Bayesian neural word embedding. In *AAAI*, pages 3135–3143, 2017.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *TACL'17*, 2017.

Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113, 2013.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Joseph Sanu, Mingbin Xu, Hui Jiang, and Quan Liu. Word embeddings based on fixed-size ordinally forgetting encoding. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 310–315, 2017.

Radu Soricut and Franz Josef Och. Unsupervised morphology induction using word embeddings. 2015.

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. Inside out: Two jointly predictive models for word representations and phrase representations. In *AAAI*, pages 2821–2827, 2016.