Learning Word Embeddings from Dictionary

Donsuk Lee
Computer Science Department
Stanford University
donlee90@stanford.edu

1 Introduction

The goal of this project to build a word embedding model that learns vector representations of new words from their English dictionary definitions. We are motivated by the observation that there is no efficient way of learning embeddings for newly coined words using current approaches. This is due to the two following facts: There are relatively few examples of new words in unlabeled text corpus. Adding an embedding for a new word usually requires re-training of models on the entire corpus , appended with examples containing the word.

Furthermore, unlike popular word embedding models, which extracts semantic and syntactic pattern of words from their co-occurrence statistics in large corpus, humans are capable of learning word meanings directly from their dictionary definition.

To this end, we have collected more than 100k word definitions from online dictionary, and trained our model on the collected dataset. We will discuss the method we used to crawl online dictionary and our model architecture in section 3 and 4.

We mainly evaluate our model's embedding on two intrinsic evaluation task: word similarity and word analogy. Furthermore, we test our model's ability to embed new words given their dictionary definitions.

Empirical results suggest that our model can i) enrich pre-trained word embeddings with the knowledge in dictionary and gain boosted performance in word similarity and word analogy task, and ii) obtain word embeddings for new words, which were unavailable during training, by looking at their dictionary definitions.

2 Related Work

Word embedding. Word embedding is the collective name for a family of methods to represent words in distributed vector space. Usually, an embedding vector encodes meaningful information about the word w that can be used to perform some natural language tasks.

Word embeddings are often trained in unsupervised fashion using large unlabeled corpus. The underlying idea is that words appearing in similar contexts must have similar meanings. Despite the absence of supervision, word embeddings exhibit useful properties; for example, words having similar meanings are mapped close to each other in the embedding space. Furthermore, initializing models with word embeddings has been shown to boost the performance in downstream NLP tasks such as syntactic parsing, sentiment analysis and question answering.

`word2vec` (Mikolov et al., 2013b) is one of the most popular neural network based word embedding models proposed by Mikolov et al. The paper introduces two model variants for learning word embeddings that are computationally less expensive than previous models: 1) Continuous bag-of-words (CBOW) and 2) Skip-gram. Word embeddings in both methods are produced as a byproduct of training neural network log-linear models on fake tasks. These tasks are not very useful by themselves, but the learned embeddings are shown to be beneficial to other tasks such as analogical reasoning.

GloVe (Pennington et al., 2014) introduces a more principled and explicit way of capturing complex relationship of words. The authors start from an intuition that the ratio of the co-occurrence probabilities of two words, rather than their co-occurrence probabilities themselves, is what contains information, and aim to encode this information as vector differences. The GloVe model learns word embeddings efficiently from the word co-occurrence statistics, while capturing complex regularities as `word2vec` does.

Using external knowledge. Although unsupervised methods produce embeddings with reasonably good quality, they may be improved by incorporating structured knowledge from already existing external resources. (Nickel and Kiela, 2017) makes use of WordNet to learn hierarchical representations of words in a hyperbolic space. (Tissier et al., 2017) incorporate co-occurrence information from dictionary definitions into the negative sampling process to move related works closer together and prevent them from being used as negative samples.

The most relevant recent work to our project is Dict2vec (Tissier et al., 2017). The paper extends the neural network architecture of Skip-gram model with negative sampling with additional context information extracted from natural language dictionary entries. However, Dict2vec treats dictionary as an auxiliary source for obtaining semantically related word pairs. Therefore, the Dict2vec model is not capable of learning word embedding for a new word directly from its definition.

One-shot learning Our work is also related to the literatures on one-shot learning, as our model can produce embeddings for new words on-the-fly, given their definition.

3 Data collection

One advantage of our approach is that collecting a dataset of word definitions is not very difficult, as many dictionaries are readily available on the Internet.

Specifically, we have collected word-definition pairs from a popular online dictionary, *Dictionary.com*. The web page containing the definitions of a `word` can be accessed easily via URL `http://www.dictionary.com/browse/[word]?s=t`. Typically, the web page shows rich linguistic information about a word, including its various meanings and part-of-speeches. In order to make full use of the available knowledge, we save each dictionary entry as a tuple of (word, part-of-speech, meaning).

The data collection is fully automated using a web crawler built with Beautiful Soup HTML parser.We can construct a dataset for arbitrary number of word-definitions without any cost as long as they exist in the dictionary.coms database. However, since our goal is to compare our model to the popular word embedding models, we queried the website with 400,000 words in GloVe 6B embeddings and downloaded definitions for 118,045 words.

Inflected words. We say a word is inflected when the form of the word is changed to convey grammatical function or attribute such as tense or quantity. Generally, an inflected word (e.g. ”higher”) is not defined separately in a dictionary, and we look for the root (e.g. ”high”) of the word. In such case, we append inflection tag to the definition of the root in order to represent the definition of the inflected form in our dictionary dataset.

4 Model

On a high level, our model learns to map definitions to the corresponding word. Given a word definition, our model outputs a vector that is close to the embedding of the corresponding word.

Formally, we define a *definition* of a word $y \in W$ as a triplet $\mathbf{x} = (p, q, w_1, ..., w_n)$, where $p \in V_{pos}$ and $q \in V_{inf}$ are part-of-speech and inflection tokens and $w_1, ..., w_n \in W$ are the word sequence describing the meaning of y. We include POS and inflection tags in the word definitions in order to disambiguate mappings to the words with the same meanings but different forms or part-of-speech.

Due to the nature of word definitions in dictionary, multiple definitions $\mathbf{x}_1, ..., \mathbf{x}_n$ can be mapped to a single word y. Conversely, a single definition may be mapped to a number of synonymous words. Such property is desirable as we want similar words to be close together in the embedding space.

Given a definition $\mathbf{x} = (p, q, w_1, ..., w_n)$, our model embeds $p, q, w_1, ..., w_n$ to vectors $\mathbf{v}_p, \mathbf{v}_q, \mathbf{v}_{w_1}, ..., \mathbf{v}_{w_n}$ in $\mathbb{R}^D$ respectively. Then, the model combines the vectors into $\mathbf{v}'_\mathbf{x} \in \mathbb{R}^D$, the final vector representation of the definition $\mathbf{x}$

Specifically, we use a bidirectional RNN (Schuster and Paliwal, 1997) with GRU cells (Cho et al., 2014) to encode encode $\mathbf{v}_p$ and $\mathbf{v}_{w_1}, ..., \mathbf{v}_{w_n}$. The hidden states of the bidirectional RNN are initialized with $\mathbf{v}_p$. Then, the sequence $\mathbf{v}_{w_1}, ..., \mathbf{v}_{w_n}$ is given as input to the network in both forward and backward directions. The last hidden layers of forward and backward RNNs are concatenated

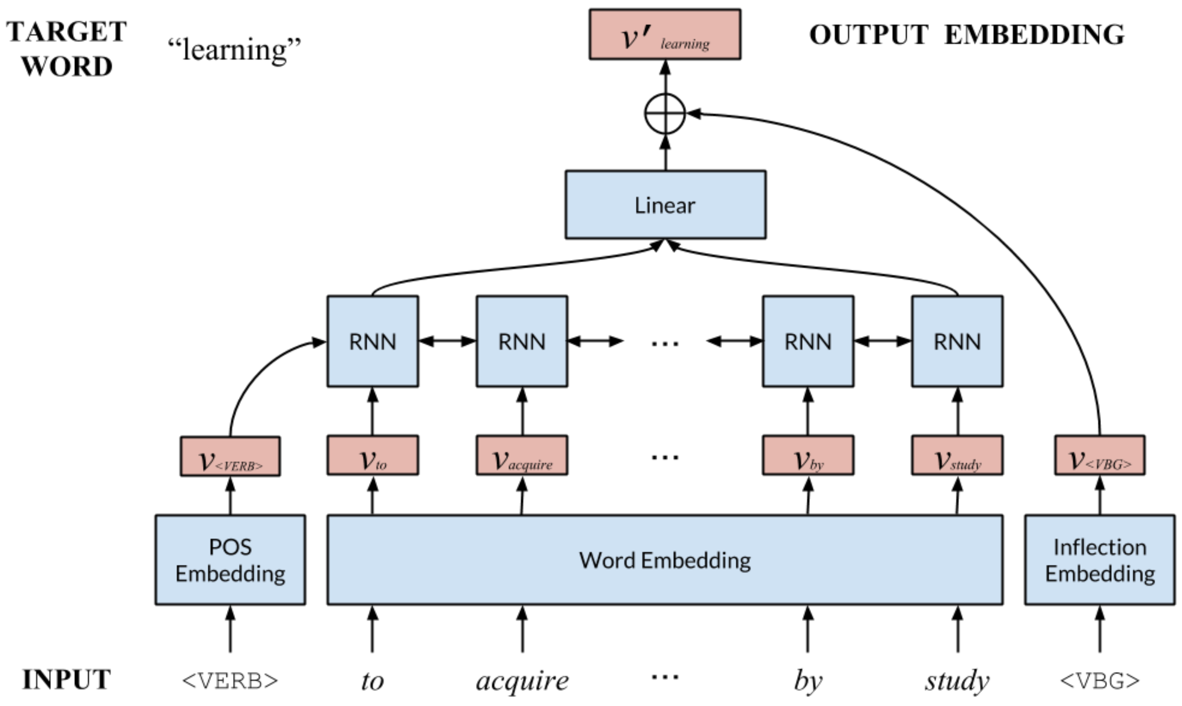

Figure 1: Model diagram

and passed through a linear layer. Finally, we add $\mathbf{v}_q$ to the output of the linear layer to obtain the final vector representation of $\mathbf{x}$. The entire model architecture is visualized in Figure 1

The POS embedding $\mathbf{v}_p$ serves as the starting point to compute the representation of the input definition $\mathbf{x}$. Furthermore, drawing inspirations from the syntactic word analogy tasks in (Mikolov et al., 2013a), we enforce our model to learn inflections as translations in word embedding space.

Learning objective. We want the output embeddings of the definition $\mathbf{x}$ to be mapped close to the word embedding of the corresponding word y. Let $\mathbf{v}'_{\mathbf{x}} = f_\theta(\mathbf{x})$ denote the output embedding given the definition $\mathbf{x}$, and let $\mathbf{v}_y$ be the word embedding of y from the model's embedding layer. Given the dataset of word-definition pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=0}^N$, we would like to minimize the quantity:

$$\frac{1}{N}\sum_{i=0}^{N} \|\mathbf{v}'_{\mathbf{x}_i} - \mathbf{v}_{y_i}\|_2 \tag{1}$$

However, this objective does not prevent the model from always outputting zero vectors for both word and definition embeddings.

We make use of pre-trained word embeddings to avoid this. We first initialize the model's embedding layer with a pre-trained embedding such as GloVe. Also, we add the following term to penalize learned embeddings for moving away from pre-trained word embeddings:

$$\frac{1}{N}\sum_{i=0}^{N} \|\mathbf{v}_{y_i} - \mathbf{u}_{y_i}\|_2 \tag{2}$$

, where $\mathbf{u}_{y_i}$ is the pre-trained word embedding for y_i. The pre-trained embeddings remain fixed during the training.

The final objective is the convex combination of (1) and (2):

$$\frac{1}{N}\sum_{i=0}^{N}(1-\lambda)\|\mathbf{v}'_{\mathbf{x}_i} - \mathbf{v}_{y_i}\|_2 + \lambda\|\mathbf{v}'_{\mathbf{x}_i} - \mathbf{v}_{y_i}\|_2 \tag{3}$$

Learned embeddings. There are two ways to extract the learned embeddings from the trained model: First, we can directly take the weights of the model's embedding layer. Second, we can average definition embeddings for each word in the training dataset. Empirically, the first method has shown better results in our experiments.

The main advantage of our model is that it efficiently obtain word embeddings for new words, which are not available during training phase. Given the definition for a new word, we can use the second method to derive its word embedding without requiring re-training of the model. Existing approaches such as (Pennington et al., 2014) or (Mikolov et al., 2013b) require re-training of models on the entire corpus , appended with examples containing new word.

5 Experiments

5.1 Training details

We used PyTorch implementation of Bidirectional RNN with GRU cells. The dimensions of the hidden layer embedding layers were set to 100 for all the experiments. We trained our model on the dictionary dataset we collected for 10 epoch using ADAM optimizer with learning rate of 0.001.

5.2 Evaluation methods

In this paper we mainly focus on two intrinsic evaluation tasks: word similarity and word analogy. We tested our model on four word similarity datasets (Halawi et al., 2012; Bruni et al., 2014;) and the word analogy dataset of Mikolov et al. (2013a)

For both tasks, we used the vector space model (vsm) libraries and evaluation scripts available on Stanford CS224U course website (`http://web.stanford.edu/class/cs224u/`).

Word similarity. Word similarity datasets consist of word pairs annotated with similarity scores judged by humans. We compute Spearman rank correlation between the human annotations and cosine distances of word pairs obtained using different word embeddings.

Word analogy. We tested on 5 syntactic (`adj-to-verb`, `opposite`, `comparative`, `present-participle`) and 2 semantic (`family`, `capital-world`) subsets of the entire word analogy dataset. Word analogy task requires finding nearest neighbors across the entire set of word embeddings. This makes evaluation quite slow especially for a large set of embeddings. In order to expedite the process, we experimented with a smaller set of 4,700 words shared by our embeddings and baselines.

Baselines. We compare our model's performance to the following baselines:

1. PPMI on `imdb-window20-flat`
2. PPMI on `gigaword-window20-flat`
3. 100D version of GloVe 6B embeddings

Embeddings for new words. In order to test the quality of embeddings for new words produced by our model, we held out a few examples from the dataset during the training for testing. After the training has completed, we obtained embeddings for the test examples by embedding their definitions using our model. Finally, we sampled the 5 nearest neighbors of each test example among the entire vocabulary. The test word-definition pairs used in our experiments are listed in Table 1.

6 Results

The experimental results for word similarity task and word analogy task are shown in Table 2 and 3

For word similarity tasks, our model achieves best performance on two of the datasets. Note that our model outperforms GloVe baseline on every dataset.

For word analogy task, our model beats baselines for both semantic and syntactic cases by a large margin.

Finally, we show the 5 nearest neighbors of the new words in 4. We can see that our model can encode both semantic and syntactic information of the new word definitions. For instance, the dictionary definition of ”hwan” is formoer monetary unit of South Korea. We can see that the 5 nearest neighbors retrieved by our model are also monetary units in different countries. On the other hand, the words retrieved by GloVe are romanizations of Korean syllables without specific meanings.

Note that the word embeddings ”polysexual” and ”unfriend” are unavailable in the GloVe 6B embeddings, so that we cannot fetch the nearest neighbors using GloVe.

7 Conclusion

References

. 2002. Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.*, 49(1):1–47.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Word		Dictionary definition
hwan	noun	a former monetary unit of South Korea.
matcha	noun	a finely ground powder made from small green tea leaves
hootch	noun	liquor illicitly distilled and distributed.
bromance	noun	a relationship or friendship between two men that is extremely close but does not involve sex.
lauzon	noun	a town in S Quebec, in E Canada, across from Quebec City on the St. Lawrence.
polysexual	adj	incorporating many different kinds of sexuality; pansexual
unfriend	verb	to remove (someone) from a list of friends or contacts on a social networking website.

Table 1: Definitions of words held out during training.

Model	MEN	MT287	MT771	WS535
PMI-i	57.2	59.9	46.2	46.9
PMI-g	64.2	**66.2**	50.9	**57.5**
GloVe	69.3	61.9	58.1	52.8
Ours	**71.9**	64.6	**61.4**	57.4

Table 2: Word similarity task

Model	Sem.	Syn.	Tot.
PMI-i	37.8	28.5	32.3
PMI-g	66.2	41.4	53.3
GloVe	89.7	63.5	76.0
Ours	**92.8**	**66.6**	**79.1**

Table 3: Word analogy task

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1406–1414, New York, NY, USA. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/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 C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc.

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.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Julien Tissier, Christopher Gravier, and Amaury Habrard. 2017. Dict2vec : Learning word embeddings using lexical dictionaries. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics.

hwan	**matcha**	**hootch**	**bromance**	**lauzon**	**polysexual**	**unfriend**
Neighboring words (ours)						
kyat	caboclo	arrack	pederastic	repentigny	pansexual	alphabetize
tugrik	japan	furfural	synergism	cowansville	songful	unbolt
nakfa	technos	rogut	cause-and-effect	chibougamau	simpleminded	nonreaders
vatu	umeboshi	non-diary	non-marital	lachute	piteous	vloggers
markka	hwan	sweetmeat	homonymy	saint-lambert	frolicsome	friended
Neighboring words (GloVe)						
tae	pu-erh	bb95	darknet	dubreuil		
chun	temi	usanee	geyi	bourzat		
hyun	rooibos	peashooter	baghead	rochette	N/A	N/A
jung	lapsang	skyjacking	i.o.u.s.a.	schoenfelder		
seok	chanoyu	flamer	rom-com	fargus		

Table 4: 5 nearest neighbors of new words.