Multi-Sense Embeddings per Word

Peng Jin

pengjing@gmail.com

Abstract

Recently, word embeddings have been used in many natural language processing problems successfully and how to train a robust and accurate word embedding system efficiently is a popular research area. Since many, if not all, words have more than one sense, it is necessary to learn vectors for all senses of word separately. Therefore, in this project, we have explored two multi-sense word embedding models, including Multi-Sense Skip-gram (MSSG) model and Non-parametric Multi-sense Skip Gram model (NP-MSSG). Furthermore, we propose an extension of the Multi-Sense Skip-gram model called Incremental Multi-Sense Skip-gram (IMSSG) model which could learn the vectors of all senses per word incrementally. We evaluate all the systems on word similarity task and show that IMSSG is better than the other models.

1 Introduction

Distributed word representations, which represent words by dense, real-valued vector embeddings, have achieved remarkable performance on several natural language processing tasks (Mnih and Hinton, 2007; Collobert and Weston, 2008; Turian et al., 2010). By placing near each other words having similar semantic

and syntactic roles, distributed word representations help address the curse of dimensionality and improve generalization.

Trained on large volumes of data, embeddings can obtain a substantial benefit. Compared to Brown clusters (Brown et al., 1992), distributed word representations have the advantages of substantially better scalability in the training and intriguing potential for the continuous and multi-dimensional interrelations. Thus, distributed word representations have been common input features for many tasks, such as named entity extraction (Miller et al., 2004; Ratnov and Roth, 2009) and parsing (Täckström et al., 2012). Passos et al. (Passos et al., 2014) train a Skip-gram model that injects supervision with lexicons and apply the obtained continuous vector embeddings to entity extraction. Bansal et al. (Bansal et al., 2014) adopt the Skip-gram embeddings to dependency parsing.

Despite the recent advances of leveraging distributed word representations, the polysemy and homonym are ignored in these representations in which each word type has only one vector representation. For example, the word *plant* has different contextual semantics relating to biology, placement, manufacturing and power generation. While the distributed representation of *plant* only has an embedding that is approximately the average of these contextual semantics. While in moderately high-dimensional spaces a vector can be relatively

“close” to multiple regions at a time, but this does not negate the unfortunate influence of the triangle inequality here: words that are not synonyms but are synonymous with different senses of the same word will be pulled together. How to fit the constraints of legitimate continuous gradations of semantics without the additional encumbrance of the illegitimate triangle inequalities still remains as an open research problem.

Some work (Huang et al., 2012; Reisinger and Mooney, 2010; Kong et al., 2019b) try to discover embeddings for multiple sense per word type via pre-clustering the contexts of a word into discriminated senses. According to these clusters, the tokens in the corpus could be re-labelled according to different senses and then learn embeddings for these re-labeled words. However, this method loses the opportunity to jointly learn the sense vectors and the clustering.

In order to tackle this problem, a multi-sense skip gram model (MSSG) is proposed in Neelakantan et al. (2015). In this method, sense vectors and context clusters are learned jointly with the assignment of token contexts to sense. In MSSG model, the number of sense per word type is fixed but in reality, some words may have much more senses than others. Therefore, it is necessary to make the number of sense per word type flexible.

In this project, we have tried two models to solve this problem and compare their performance on word similarity task.

- Non-parametric Multi-sense Skip Gram model (NP-MSSG) (Neelakantan et al., 2015). In this model, the number of clusters (sense) is flexible and a new cluster (sense) will be added according to distance of its context to the nearest cluster (sense).
- Incremental Multi-sense Skip Gram model (IMSSG). In this model, we adapt the original MSSG, which could increase

the number of sense of a word incrementally and stop increasing the number of sense of a word if there exists two very similar clusters (sense) already.

Our report is structured as follows. We first overview the recent research on word representation learning in Section 2 and introduce Skip-gram model in Section 3, we then present our proposed model in Section 4, followed by a description of experiments and results in Section 5. We conclude and present future directions in Section 6.

2 Related Work

Bengio et al. (2012; 2016) propose to extend the traditional idea of n -gram language models by replacing the conditional probability table with a neural network, representing each word token by small vector instead of an indicator variable and estimating the parameters of the neural network and these vectors jointly. To reduce the expensive computation in Bengio et al.’s work (2012), Collobert and Weston (2008) replace the max-likelihood character of the model with a max-margin approach where the network is encouraged to score the correct n -grams higher than randomly chosen incorrect n -grams. Mikolov et al. (2013a) and Mikolov et al. (2013b) propose extremely computationally efficient log-linear neural language models by removing the hidden layers of the neural networks and training from larger context windows with very aggressive subsampling.

Despite the maturity of the research on learning vector representations of words, there is relatively less prior work on learning multiple vector representations for the same word type. Reisinger and Mooney (Reisinger and Mooney, 2010; Kong et al., 2019a) propose to construct multiple sparse, high-dimensional vector representations of words. Huang et al. (Huang et al., 2012) extend this approach incorporating global document context to learn

Figure 1: Architecture of the Skip-gram model with window size $R = 2$. Context c_i of word w_i is $w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}$

multiple dense, low-dimensional embeddings by using a recursive neural network.

3 Skip-gram model

The Skip-gram model learns word embeddings through predicting the surrounding words in a sentence. In the Skip-gram, $v(w) \in R^d$ is the embedding vector for the word w where d is the dimension of the embedding. In more details, given a pair of words (w_i, w_j) , the probability that the word w_j is observed in the context of word w_i is given by:

$$P(D = 1|v(w_i), v(w_j)) = \frac{1}{e^{-v(w_j)^T v(w_i)}} \quad (1)$$

Also, the probability of not observing word w_j in the context of word w_i is $1 - P(D = 1|v(w_i), v(w_j))$. Therefore, given a training set including the sequence of words $w_1, \dots, w_T$, the word embeddings are learned by maximiz-

ing the following objective:

$$J = \sum_{(w_j) \in c_i^+} \log P(D = 1|v(w_i), v(w_j)) + \sum_{(w'_j) \in c_i^-} \log P(D = 0|v(w_i), v(w'_j)) \quad (2)$$

where w_i is the i -th word in the training set, c_i^+ is the set of context words of word w_i and c_i^- is the set of randomly selected context words for the word w_i . In more details, the set of context words for a given word w_i is $c_i^+ = \{w_{i-R}, \dots, w_{i-1}, w_{i+1}, w_{i+R}\}$, where R is window size. A sample of the Skip-gram model is shown in Fig. 1

4 Methodology

4.1 Multi-Sense Skip-gram (MSSG) model

In the MSSG model, each word w has a global vector $v_g(w)$ and there is an embedding $v_s(w, k)$ and a context cluster $\mu(w, k)$ ($k = 1, \dots, K$) for each sense of the word, where K is a hyperparameter. Note that K is universal across all words which means that the number of sense for each word is the same. The global vector $v_g(w)$ is the average vector of all sense vectors.

Just like the Skip-gram model, in the MSSG model, given a word w_i , we could obtain the context of this word $c_i = \{w_{i-R}, \dots, w_{i-1}, w_{i+1}, w_{i+R}\}$, then we could calculate the vector representation of the context c_i , i.e., $v_{context}(c_i) = \frac{1}{2 \times R} \sum_{w_j \in c_i} v_g(w_j)$. Note that here we use the global vector instead of its sense vectors of the context words to avoid the high computational cost. After getting the context vector presentation, we could predict the sense of the word w_i , s_i according to:

$$s_i = \underset{k=(1, \dots, K)}{\operatorname{argmax}} \operatorname{sim}(\mu(w_i, k), v_{context}(c_i)) \quad (3)$$

Figure 2: Architecture of the Multi-Sense Skip-gram model with window size $R = 2$ and the number of sense $K = 3$. Context c_i of word w_i is $w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}$. The sense of the word w_i is predicted by finding the cluster vector of the context which is nearest to the average of the context vectors.

Intuitively, we calculate the similarity between context vector and each cluster of word w_i and choose the nearest one as the predicted sense of the word w_i in this context. Here cosine similarity is used in experiments. After predicting the sense of word w_i , the context cluster vector $\mu(w_i, s_i)$ is also updated since context c_i is added to s_i context cluster, we also could calculate the probability of a word w_j observing in the context of word w_i by:

$$P(D = 1 | v_s(w_i, s_i), v_g(w_j)) = \frac{1}{e^{-v_g(w_j)^T v_s(w_i, s_i)}} \quad (4)$$

Similar to the Skip-gram model, the objective function of the MSSG model is:

$$J = \sum_{(w_j) \in c_i^+} \log P(D = 1 | v_s(w_i, s_i), v_g(w_j)) + \sum_{(w'_j) \in c_i^-} \log P(D = 0 | v_s(w_i, s_i), v_g(w'_j)) \quad (5)$$

where c_i^+ is the set of context words and c_i^- is the set of noisy words for the word w_i .

4.2 Non-Parametric MSSG model (NP-MSSG)

One major problem of the MSSG model is that the number of sense for each word is fixed which is K . However, some words may have more than K senses and other words have less than K senses. The NP-MSSG model is able to learn various number of sense per word. Similar to the MSSG model, each word w is associated with sense vectors, contexts clusters and a global vector. Different from the MSSG model, the number of sense vectors and context clusters is unknown at first. The first sense vector and context cluster for each word will be created on its first occurrence in the training data. After creating the first context cluster for a word, a new context cluster and a sense vector are created online during training when

the similarity between the vector representation of the context with every existing cluster center of the word is less than λ , where λ is a hyperparameter of the model. Therefore, the major difference between NP-MSSG and MSSG is the process of sense selection. In the sense selection of the NP-MSSG model, we also have to calculate the context vector $v_{context}(c_i) = \frac{1}{2*R} \sum_{w_j \in c_i} v_g(w_j)$ first. Let $k(w_i)$ be the number of context clusters currently associated with the word w_i . s_i , the sense of word w_i is given by:

$$s_i = \begin{cases} k(w_i) + 1 & \text{if } \max_{k=1, \dots, k(w_i)} \{\text{sim}(\mu(w_i, k), v_{context}(c_i))\} < \lambda \\ k_{max} & \text{otherwise} \end{cases} \quad (6)$$

where $\mu(w_i, k)$ is the cluster vector of the k -th sense of the word w_i and $k_{max} = \text{argmax}_{k=1, \dots, k(w_i)} \text{sim}(\mu(w_i, k), v_{context}(c_i))$.

The cluster center is the average of the vector representations of all the contexts which belong to that cluster. If $s_i = k(w_i) + 1$, a new context cluster and a new sense vector are created for the word w_i .

In summary, the NP-MSSG model and the MSSG model described previously differ only in the way that word sense discrimination is performed. The objective function and the probabilistic model associated with observing a (word, context) pair given the sense of the word remain the same.

4.3 Incremental Multi-Sense Skip-gram model (IMSSG)

In the IMSSG model, each word w is associated with sense vectors, contexts clusters and a global vector. In the MSSG model, the number of sense per word, K , is fixed before training. However, in the IMSSG model, we assume that there is only one sense for each word at first, therefore, we could train the current system by the MSSG model with $K = 1$ (since the number of senses per word is fixed in the current training round). After training, we will check the context clusters for each word. If there are

two context clusters too similar, i.e., the similarity between these two context clusters are more than ϕ which is a hyperparameter, we will think that we have found all senses of this word in the training dataset and stop increasing the number of senses for this word. If not, we will increase the number of sense for this word by one and retrain the system. The following equation could clarify this process:

$$k(w_i) = \begin{cases} k(w_i) & \text{if } \exists(j, z) \{ \text{sim}(\mu(w_i, k_j), \mu(w_i, k_z)) < \phi \\ k(w_i) + 1 & \text{otherwise} \end{cases} \quad (7)$$

We also set the largest number of sense for each word, which is 10 in our experiment, which means the largest number of senses for each word in our experiment is 10.

Except the sense discrimination process, the other training processes and objective function remain the same as the MSSG model.

5 Experiments

5.1 Experimental Setup

In this section, we present our experiment setups by introducing the benchmarks, the experiment settings and evaluation metrics used for evaluation.

Implementation details We train all our models using clean Wikipedia text corpus text8 and important hyper parameters in our models are declared in Table. 1.

Datasets

The text8 corpus is a collection of clean Wikipedia text. There are 253,855 unique words and the total number of words is 17,005,208. We train our embedding system on this dataset by different methods.

We use the word similarity task to evaluate our different models and the following are the dataset we used in experiments:

The WordSim-353 (Finkelstein et al., 2001) database is a standard dataset for evaluating word vector representations. It consists of a list of pairs of word types, the similarity of which

Parameter Name	Value
Vocabulary size	100,000
embedding dimension	300
Window size	5
Sense number K in MSSG	3
λ in NP-MSSG	-0.5
ϕ in IMSSG	0.7

Table 1: Hyperparameters of models

is rated in an integral scale from 1 to 10. Pairs include both monosemic and polysemic words. These scores to each word pairs are given without any contextual information.

The Stanford’s Contextual Word Similarities (Huang et al., 2012) (SCWS) database consists of 2003 word pairs and their sentential contexts. It consists of 1328 noun-noun pairs, 399 verb-verb pairs, 140 verb-noun, 97 adjective-adjective, 30 noun-adjective, 9 verb-adjective, and 241 same word pairs.

The Stanford Rare Word Similarity (RW) Dataset focuses on rare words to complement existing ones. In this dataset, rare words will be paired with other words to test the robustness of an embedding systems. There are 2034 word pairs in this dataset.

Word Similarity We choose word similarity as the task to evaluate our models. The Spearman correlation between a model’s similarity scores and the human judgements in the datasets is reported and basically, the higher the Spearman correlation score is, the better the model is.

Nearest Neighbors Besides the word similarity task, we also find the nearest neighbors of a word given different models. The nearest neighbors of a word are computed by comparing the cosine similarity between the embedding for each sense of the word and the global embeddings of all other words in the vocabulary. Here we report the three most nearest neighbors of the word “apple”.

5.2 Results and Discussion

5.3 Word Similarity

We report the Spearman correlation between a model’s similarity scores and the human judgements in Table. 2. We could see that IMSSG could achieve the best performance on the WS353 and SCWS datasets and MSSG is better than other models on RW dataset. NP-MSSG and IMSSG model is generally better than the MSSG model. When we compare the results between IMSSG and NP-MSSG, the result of IMSSG is better on all of the datasets.

5.4 Nearest Neighbors

We report three most nearest neighbors of the word “apple” in the Table. 3. For the Skip-gram model, it is easy to find the sense is mixed up since “macintosh” and “strawberry” are basically two different things. For the MSSG model, it is much better than the Skip-gram model since it is clear to find the first sense is fruits and the second and the senses are technique products and technique companies although it mix up the products and companies. When we check the result of NP-MSSG, there are only two senses, one is fruit and the other is technique products. The result of IMSSG is the best to us since it could separate the technique products and technique companies perfectly. Specifically, in the second sense, the nearest neighbors are all companies and in the third sense, they are all products.

5.5 Comparison between NP-MSSG and IMSSG models

NP-MSSG is a dynamic and non parametric model, which means that the number of sense could be dynamically changed during the training but in IMSSG, we have to increase the number of sense per word after a training round. From the results, it is clear to see that IMSSG is a little better than NP-MSSG, we think this is because IMSSG is more stable than NP-MSSG. In another word, IMSSG increases the number of sense for each word after a train-

	WS353	RW	SCWS
Skip-gram	0.671	0.313	0.611
MSSG	0.684	0.337	0.624
NPMSSG	0.689	0.324	0.637
IMSSG	0.698	0.332	0.651

Table 2: Results of different models on word similarity tasks. The numbers are Spearman correlation between each mode’s similarity scores and the human judgements Best result on each dataset are in bold face.

Model	nearest neighbor words
Skip-gram	strawberry, blackberry, macintosh
MSSG	strawberry, blackberry, plum microsoft, macintosh,iphone pc, ibm,smartphone
NP-MSSG	strawberry, pear, plum macintosh, iphone, smartphone
IMSSG	strawberry, blackberry, pear microsoft, intel, ibm pc, smartphone, iphone

Table 3: Nearest Neighbors of the word apple for different models

ing round and at this time, the embedding for each word is more accurate.

6 Conclusion

In this projet, we explore multi-sense word embedding models which are Multi-Sense Skip-gram model and Non-Parammetric Multi-Sense Skip-gram model and present an extension of Multi-Sense Skip-gram model called Incremental Multi-Sense Skip-gram model which could learn multiple embeddings per word type with varying number of senses incrementally and jointly learn word sense discrimination and word embeddings at the same time. We evaluate all models using word similarity task and the proposed model is better than the other models. In the future, we will apply our model on larger text corpus and evaluate it on other external tasks.

References

- Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *ACL*.
- Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18(4):467–479.
- Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.
- Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*.
- Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*.
- Xiang Kong, Preethi Jyothi, and Mark Hasegawa-Johnson. 2016. Performance improvement of probabilistic transcriptions with language-specific constraints. *Procedia Computer Science* 81:30–36.
- Xiang Kong, Zhaopeng Tu, Shuming Shi, Eduard Hovy, and Tong Zhang. 2019a. Neural machine translation with adequacy-oriented learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*. volume 33, pages 6618–6625.
- Xiang Kong, Qizhe Xie, Zihang Dai, and Eduard Hovy. 2019b. Fast and simple mixture of softmaxes with bpe and hybrid-lighttrnn for language generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. volume 33, pages 6626–6633.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *ICLR*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*.
- Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT-NAACL*.

- Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *ICML*.
- Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*.
- Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *CoNLL*.
- Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. ACL.
- Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *HL-ACL*.
- Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*. ACL.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*.