Using Embedding Masks for Word Categorization

Stefan Ruseti, Traian Rebedea and **Stefan Trausan-Matu**
University Politehnica of Bucharest
{stefan.ruseti, traian.rebedea, stefan.trausan}@cs.pub.ro

Abstract

Word embeddings are widely used nowadays for many NLP tasks. They reduce the dimensionality of the vocabulary space, but most importantly they should capture (part of) the meaning of words. The new vector space used by the embeddings allows computation of semantic distances between words, while some word embeddings also permit simple vector operations (e.g. summation, difference) resembling analogical reasoning. This paper proposes a new operation on word embeddings aimed to capturing categorical information by first learning and then applying an embedding mask for each analyzed category. Thus, we conducted a series of experiments related to categorization of words based on their embeddings. Several classical approaches were compared together with the one introduced in the paper which uses different embedding masks learnt for each category.

1 Introduction

The idea of using vector representations of words for various natural language processing (NLP) and machine learning tasks has become more and more popular in the last years. Most of these representations are based on the idea that the meaning of a word can be determined by the context in which each word is used.

Sometimes, additional information about the words is available or can be computed and this might be used along with the embedding for each word. This information may consist of relations between words (e.g. syntactic dependencies), part of speech (POS) tags, word categories, word senses, etc.

In this paper, we propose to encode this extra information in the form of a vector mask that can be applied on the word embedding before being used as an input by any classifier, such as a neural network, or before computing any semantic distance between the word embeddings. We explore the possibility of using vector masks for assigning WordNet (Miller, 1995) categories to words. We define a word category as one of the top concepts in the WordNet taxonomy as will be later explained in more detail. Using the trained masks for a subset of words, we then test whether they improve the accuracy of determining the correct category for new words that are not part of the training corpus.

2 Related Work

Distributed words' embeddings based on word co-occurrences can be computed using various mechanisms and theories. Some of them employ algebraic decompositions of the original vector space, others use mixture models to compute a distribution of words in topics from a large collection of texts, while newer methods make use of neural embeddings to train word representations on even larger corpora of texts than the previous models. All the methods described in this section are completely unsupervised and are based on the frequency of words in documents and their co-occurrences.

Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997) is a commonly used vector representation for words. The word vectors are obtained through a Singular Value Decomposition (SVD). The main reasoning behind LSA is that the decomposed space can generalize the relationships between words and documents existing in the original term-document matrix and will remove noisy features. While SVD is generally used for LSA,

Proceedings of the 1st Workshop on Representation Learning for NLP, pages 201–205,
Berlin, Germany, August 11th, 2016.

other matrix factorizations such as Non-negative Matrix Factorization (Lee et al., 2010) have been successfully employed for various tasks.

A newer approach which can also be used for computing embeddings is Latent Dirichlet Allocation (LDA) (Blei et al., 2003). This model assumes that each document is a mixture of topics and each topic is defined as a distribution over the words in the corpus. Although LDA is mainly used for topic modelling, it can also be employed for computing word embeddings, which are represented by the probabilities that each word is included in a topic.

One of the most recent and popular models for training word embeddings is Word2Vec (Mikolov et al., 2013a) which makes use of neural embedding models. The word representations are computed by a neural network that predicts the probabilities of a word occuring in a context window. Levy and Goldberg (2014) showed that this model performs, in fact, a factorization of a word-context matrix. The main advantage of this model over other similar ones is the fact that it can be trained on much larger texts, which could produce better embeddings.

GloVe (Pennington et al., 2014) is a word representation model based on the global word co-occurrence matrix which is reduced to a lower-dimensional representation after normalization and log-smoothing. The model achieves better results than Word2Vec, at least for the word analogy, word similarity and named entity recognition tasks presented in the paper (Pennington et al., 2014).

Recent work was also focused on improving given vector representations. Usually, the vectors are computed with an unsupervised method and a post-processing step is applied on the final vectors that do not depend on the representation model. Mrki et al. (2016) use pairs of synonyms and antonyms to bring words closer or apart, while keeping as much of the original topology as possible. A similar method, that uses WordNet relations between words, was proposed by Faruqui et al. (2014).

Tsubaki et al. (2013) proposed a method of improving vectors in the training step. For each word and syntactic relations, a group of frequent words was computed. Based on them, the representation of a word can be projected in another space depending on the words connected to it.

All representations presented so far learn, for simplicity, only one embedding vector per word. Neelakantan et al. (2014) propose a Multi-Sense Skip-gram that learns different representations for each sense of a word. The advantage over other multi-sense representations is the fact that sense discrimination and embedding learning are performed in the same step.

3 Problem Description

In our approach we considered WordNet for extracting word categories. WordNet contains sets of synonym words (synsets) with various linguistic relations between them. In our case, the important relations are hyponymy and hypernymy, which describe specializations/generalizations between concepts. These relations form a tree, with "entity" as the top concept in the case of nouns.

Our experiment starts from the assumption that word embeddings should partially capture the hypernym/hyponym relations from Wordnet. To test this supposition, we decided to split the nouns synset tree into several top-level subtrees rooted in concepts who subsum a somehow similar size set of words. These top-level concepts denote the word categories. We opted to use only a small number of categories, because of their hierarchical structure of the tree. In order to determine a balanced set of categories, a top-down approach was used where a candidate concept for a category was split if its subtree contained too many words or if it had more than one child containing most of the words in that subtree.

One of the problems that we encountered was the fact that there is a many-to-many relation between words and synsets. Since the hierarchy was based on synsets and the embeddings are computed on words, a simply greedy approach was chosen by taking the first word for each synset.

In the filtering process, which established the initial balanced categories for our experiment, we kept only the synsets corresponding to single word lemmas that had a corresponding vector in our pre-trained Word2Vec representations.

After computing the mapping between words and categories, a corpus was built in order to test the hypothesis that embeddings can be used to determine the category of a word. The generated dataset consists of triples of the form $(word, category, result)$, where $result$ is 1 when the first synset of the $word$ is part of $category$ and 0 otherwise. For simplicity, the number of positive and negative examples in the dataset is equal.

4 Vector Operations and Masks for Word Embeddings

We start from the more complex assumption that two adjacent words in a sentence should have a combined meaning depending on their individual senses and the type of syntactical dependency between them. If a mathematical function may be learned for each dependency type, then the meaning of the sentence may be recursively computed by combining embeddings two by two.

The word embeddings computed using the skip-gram method presents another interesting feature. Mikolov et al. (2013b) showed that vectors can be combined to resemble analogies. A famous example is "$King - Man + Woman = Queen$", where the operations are applied on the corresponding vectors of each word, and the closest vector to the result is the one corresponding to the word *Queen*. We hope to detect other relations between words, relations that could be expressed by more complicated mathematical functions in the embedding space.

Assuming that we use word embeddings of size *d*, we have to find a function that combines the two vectors and the dependency to produce another vector of size *d*. Considering that the dependency also uses some embedding, of size *d'*, the function to generate the combined embedding can be expressed like $W_{d,2d+d'} * [w_1; w_2; dep]$. The problem with this representation is that it cannot capture relations similar to $w_1 - w_2$ which exist in the vector space. Thus the dependency should not be added as a distinct feature, but rather define how the two embeddings are combined.

The simplest solution is to consider different networks for each dependency type. This would allow us to represent any function between the words, but requires a very large number of weights. This might not be possible due to the limited training examples and because some dependency types are too rare.

In order to decrease the number of required weights, the dependency can be represented as a mask. The mask can be applied as a point-wise multiplication with the $[w_1; w_2]$ vector, which allows learning transformations like $w_1 - w_2$.

In a first experiment, we decided to test the vector masks on the word categorization dataset. The assumption is that some part of the representation of words in the same category might be common, while the other corresponds to specific context for each word. This means that two words can become closer in the vector space by neutralizing the specific dimensions for that category. In order to do this, we computed an embedding for each category in the dataset having the same size with the word embeddings. These category embeddings are used as a mask, multiplying them pointwise with a word embedding. This operation is actually a scaling in the word embedding space which should cluster the words from the same category.

5 Experiments

The solutions below were tested in similar conditions on the generated dataset. A brief description of each method was added when needed. Most of the models (Support Vector Machines - SVM, random forest and logistic regression) were developed in Weka [1], while the neural networks were implemented in Tensorflow[2].

Cosine Similarity

A common way of comparing embeddings is using the cosine similarity. A threshold can be used as a boundary between positive and negative examples. For word categorization, the best threshold was chosen based on the training set.

Multilayer Perceptron (MLP)

The network consists of one hidden layer with 100 neurons and an softmax output layer with 2 neurons representing the probabilities for the two classes. The tanh activation function was used and a dropout with probability 0.4 on the hidden layer was added to reduce the effect of overfitting. For training, we opted for cross-entropy loss function and Adagrad Optimizer for 500 epochs.

Cosine with Vector Mask

Given a word and a category, the embedding of the category is applied as pointwise multiplication both to the embedding of the word and the word depicting the category. The resulting vectors are then compared using cosine. The following loss function was used during training the vector masks:

$$max((y - y^{'})^2 - 0.25, 0) + \alpha * \sqrt{\frac{\sum_w w^2}{noweights}} \quad (1)$$

, where:

[1] http://www.cs.waikato.ac.nz/ml/weka/

[2] https://www.tensorflow.org/

- y the output of the network (the cosine similarity between vectors, normalized to 0-1)
- $y^{'}$ the target value (0 for negative and 1 for positive examples)
- w a parameter from the network (a value in the category embedding matrix)

The first term tries to achieve a maximum 0.5 difference between the output and the target value, while the second term is a regularizer to avoid overfitting.

Mask + MLP

This network combines the mask embeddings with the same MLP described earlier. First, the word and category embeddings are transformed by applying the mask. The resulting vectors are used as inputs for the MLP.

6 Results

The described methods are compared on the generated dataset for word categorization based on WordNet. The accuracy scores from Table 1 were obtained using 10-fold cross-validation.

Method	Accuracy (%)
Cosine	60.59
SVM	59.69
Logistic regression	60.12
Random forest	77.79
MLP	83.00
Mask + cosine	77.00
Mask + MLP	**85.50**

Table 1: Accuracy of the compared methods.

The poor results obtained by the SVM are not very surprising. While the words from the same category might be grouped together in the Word2Vec embedding space, the training examples consisted in pairs of words and categories. Positive pairs have no reason to be close to each other, meaning that the positive and negative examples are not linearly separable in this case.

Comparing cosine similarity with the mask-cosine method, an important improvement of 27% can be observed. This demonstrates that scaling both the word and its category will make them closer, while scaling a word and a different category will make them more distant. This means that each mask successfully minimizes the effect of dimensions in the space that are not related to the given category.

It was also expected that the mask+MLP approach to work better than a simple MLP because the first one has more parameters. In our experiments, the improvement was not impressive (about 3%), but we observed a much faster training rate. While the MLP needed 500 epochs to reach this accuracy, the version with masks achieved the same performance in 50 epochs. The masks accelerate training, but also have a tendency to quickly over-fit on the training data.

7 Conclusion

Mask embeddings proved to be useful for the proposed word categorization task. Although this is an artificial task (the category of each word is already known from WordNet), the method can be applied in other scenarios. The results show that such masks can learn which dimensions are important in a given situation. Also, the masks can be learnt much faster than a regular fully-connected layer.

An alternative for masks would be the use of different networks for each category. This solution is more general than the method proposed in this paper which uses embeddings masks and MLP; for this reason they will have more parameters than the Mask+MLP technique. Further testing is needed on this subject, but masks seem a viable solution for limiting the number of parameters of a network, which can be crucial when dealing with small datasets.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, mar.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2014. Retrofitting Word Vectors to Semantic Lexicons. nov.

Thomas K Landauer and Susan T. Dumais. 1997. A solution to Plato ' s problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104(2):211–240.

Hyekyoung Lee, Jiho Yoo, and Seungjin Choi. 2010. Semi-Supervised Nonnegative Matrix Factorization. *IEEE Signal Processing Letters*, 17(1):4–7, jan.

Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pages 1–12.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. *Proceedings of NAACL-HLT*, (June):746–751.

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

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting Word Vectors to Linguistic Constraints. mar.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew Mccallum. 2014. Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. *Emnlp-2014*, pages 1059–1069.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Masashi Tsubaki, Kevin Duh, Masashi Shimbo, and Yuji Matsumoto. 2013. Modeling and Learning Semantic Co-Compositionality through Prototype Projections and Neural Networks. In *EMNLP*, pages 130–140.