Evaluation of Word Embeddings for Sequence Tagging

A Anonymous **B Anonymous**

Abstract

XXX

1 Introduction

Word embedding learning methods are the new generation of distributional semantics models. As with other learning methods, it is well known that the performance of machine learning algorithms heavily depends on their parameters optimization, the training data used and the applications they target. In this paper, we perform an extensive evaluation of three word embedding approaches under the same experiment conditions, and evaluate them in different sequence tagging tasks.

Expected contributions:

- Fair comparison of different word embedding algorithms
- Influence of word embeddings in the sequence tagging tasks with semi-supervised settings.
- Complete evaluation of word embeddings in sequence tagging tasks.
- Multi-word expressions identification using word embeddings.

2 Background

In this section, we describe the three word embedding approaches we compared in the paper.

Explain what are word embeddings Word embeddings models are the new generation of distributional semantics models (DSMs), where the vectors weights are set to optimally predict the contexts in which the corresponding word tend to appear.

Describe each of the following word embeddings approaches

- Glove (Pennington et al., 2014)
- word2vec (Mikolov et al., 2013)
- Brown cluster (Brown et al., 1992)
- Pre-trained word embeddings (Turian et al., 2010)

The mentioned methods were chosen because they are recent state-of-the-art word embedding learning and because their their software is available.

3 Materials

We test the effectiveness of different word embeddings approaches on four different sequence labeling task, which are the following,

- POS tagging for both twitter and WSJ.
- Chunking on the CoNLL shared task data.
- MWE identification with Nathans corpus
- Named entity recognition on CoNLL shared task data and MUC7.

It is well known that the choice of a corpora have an important effect in the final accuracy of machine learning algorithms. Thus, we select different corpora to learn the word embedding vectors (Table 3). The main reason of choosing these data set is that they are publicly available.

Data set	Size	Words
One Billion	4.1GB	0.8 billions
UMBC	48.1GB	3 billions
First billion wikipedia	1GB	1 billions
Latest wikipedia dump	49.6GB	3 billions
Twiter		

Table 1: Corpus used to learn word embeddings

4 Experiments setups

4.1 Preprocessing

In order to make the comparison of different word embedding approaches across different applications, we applied the same preprocessing to the data sets used. The preprocessing pipeline consist of a sentence splitter, a tokenizer, a POS-tagger and a lemmatizer. The pipeline is built with the UIMA architecture and the DKPro NLP tools.

4.2 Parameters

The performance of each approach heavily depends on their parameters optimization. In an ideal machine learning setup, grid search or random search would be applied in order to search for the best hyperparameters, for each approach. But, that is too time taken. Instead, we look for the shared parameters along the three approaches and vary these parameters deterministically? The rest of the parameters (the ones that are unique for each approach) are set to their optimal reported values. The parameters that we vary are:

- **Word vector size**: 25, 50, 100, 200, 300, 600.
- **Context window size**: 5, 10, 15.
- **Training size**: +100 document.

The research question we want to answer with these experiments are: How does the size of training data, the vector dimensionality and the context window influence the performance of sequence tagging?

Explain why we chose this parameters, why they are important?

As pointed out by (**?**), there is dependency between the size of training set and the vector size, in the sense that the performance does not improve when only one of these variables is increased, but when both are increase in parallel. Therefore, we measure the accuracy by incrementally augmenting the size of the training set at the same time as the word vector size.

There are also some variations in how the learning algorithm used the word embedding vectors. In our experiments, we defined four different setup, which are:

1. CRF with random initialized word vectors.
2. CRF with pre-trained word vectors but not update
3. CRF with pre-trained word vectors and fine-tune the word vectors during training.
4. use all features from state-of-the-art sequence tagging models such as NICTA NER (like gazetteers).

why this variations matters? what we are expecting to happen? The research question associated with it is: Could pre-trained word vectors improve the sequence tagging if there are unknown words in the test datasets? Trained on Penn Treebank and test on English tree bank Trained on CoNLL and test on MUC7

5 Results

- Performance curves in the semi-supervised learning for each sequence tagging task.

6 Discussion

Acknowledgments

Anonymised
Anonymised
Anonymised
Anonymised
Anonymised
Anonymised

References

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*, 18:467–479.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Joseph Turian, Dpartement Dinformatique Et, Recherche Oprationnelle (diro, Universit De Montral, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semisupervised learning. In *In ACL*, pages 384–394.