Evaluation of Word Embeddings for Sequence Labelling Tasks

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 parameter optimization, the training data used and the applications they target. In this paper, we perform an extensive evaluation of four word embedding approaches under fixed experiment conditions, and evaluate them over different sequence labelling tasks.

Expected contributions:

- Fair comparison of different word embedding algorithms. This includes running different word embeddings algorithms under controlled conditions, for example, chose algorithms and data sets that are publicly available, apply the same NLP pre-processing to all data sets, trained the algorithms with the same data sets, apply the learned word vectors to well defined tasks sequence labeling tasks such as POS-tagging and Named Entity Recognition.

- Influence of word embeddings in the sequence tagging tasks with semi-supervised settings. We considered the empirical evaluation in a semi-supervised setting because we conjecture that the word embeddings learned from unlabelled data could save a significant amount of labelled data, and thus, alleviate the cost of human annotation. See Guo et al., "Revisiting Embedding Features for Simple Semi-Supervised Learning" (http://ir.hit.edu.cn/~jguo/papers/emnlp2014-semiemb.pdf).

- Comprehensive evaluation of word embeddings in sequence labelling tasks. Sequence labelling tasks is one of the meta problems NLP faced. Previous works have shown that learning word embeddings vector features improves the performance of sequence labelling task, but not fair and extensive comparison has been done so far. In this paper, we evaluated word embedding algorithms in several sequence labelling tasks and discuss their benefits, taking into account different aspects of sequence labelling problems such as row labelling (POS-tagging) and join segmentation and labelling (Named Entity Recognition).

- Multiword expressions (MWE) identification using word embeddings. To the best of our knowledge know, current work only applies Brown clustering to identify MWE. The experiments carry out in this paper will tell which is the best word embedding algorithm to solve for the identification of MWE.

Describe each of the following word embeddings approaches

- Glove (Pennington et al., 2014)

- word2vec (Mikolov et al., 2013)

- Pre-trained word embeddings (**?**)

- Brown cluster (Brown et al., 1992)

The first three methods were chosen because they are recent state-of-the-art word embedding methods and because their software is available. The final method (Brown clusters) was selected as a benchmark word representation which makes use of hard word clusters rather than a distributed representation.

2 Related Work

In the last years, distributed word representations have been applied to several NLP tasks. Their attractiveness relies in the ability to learn word representations in an unsupervised way, thus directly providing lexical features from big amounts of unlabelled data. They have been found specially useful for connection out of vocabulary words to known ones, and for encouraging common behaviour among related in-vocabulary words.

Collobert et al. (2011) proposed a neuronal network architecture that learn word embeddings and use them in POS-tagging, chunking, NER and SRL. They induced embeddings from a Wikipedia corpus of 631 millon words, with 50 dimensions, over a window size of 5 grams. Other hyperparameters they optimize were: capital feature dimension, number of hidden units, and learning rate = 0.01. Without specializing their architecture for the mentioned tasks, they achieve close state-of-the-art performance. After including specialized features (e.g., word suffixes for POS-tagging; Gazzeters for NER, etc.) and other tricks like cascading and ensambling classifiers, achieve competitive state-of-the-art performance.

Similarly, Turian et al. (2010) explored the impact of using word features learned from cluster-based and word embeddings representations for NER and chunking. They evaluate Brown clusters, and word embeddings from Collobert and Weston (2008) and Mnih and Hinton (2008). Words representations were learned using the same data set (a sub-sample of the of the RCV1 corpus), of 37 millions words and a vocabulary size of 269K. They induced embeddings with 25, 50, 100, and 200 dimensions over 5-gram windows and 50 epochs for the algorithm from (Collobert et al., 2011); 50 and 100 dimensions over 5-grams windows and 70 epochs for the algorithm from (Mnih and Hinton, 2008); and clusters solutions of 100, 320, 3200 clusters. According to their experiments, the best scaling parameter, is 0.1. They conclude that unsupervised word representation improve NER and chunking, and that combining different word representations can further improve the performance.

Owoputi et al. (2013) demonstrated that Brown clusters enhance Twitter POS tagging. They induced 1000 clusters using a data set of 847 millions of tokens (from 56 million unique tweets). Only units that appeared with a frequency higher than 40 were clustered, which results in a vocabulary size of 216,856.

Schneider et al. (2014a) presented a MWE analyzer that, among other features, used unsupervised word clusters. Using words that appears at least 20 times in the Yelp Academic Dataset (21 million words), they induced 1000 Brown clusters. They conclude that the clusters were useful for identifying words that usually belong to proper names. Nevertheless, they mentioned that it is difficult to measure the impact of the word embeddings features, since other features may capture the same information.

Bansal et al. (2014) used word embeddings as features for dependency parsing. To built the vectors, they used the syntactic dependency context instead of the linear context in raw text. They found that simple attempts based on discretization of individual word vector dimensions do not improve parsing. Only when performing hierarchical clustering of the continuous word vectors then using features based on the hierarchy, they gain performance. They also pointed out that ensemble of different word embeddings representations improved performance. An interesting observation the point out is that with large of window size words tend to topically group, and with small window size words tends to share the same POS-tag. Within the same aim, (Andreas and Klein, 2014) explores the used of word embeddings for constituency parsing. They mentioned that even word embeddings contain useful syntactic information, this information might be redundant with the one acquire by a syntactic parser trained with a small amount of data. Their intuition is that, meanwhile word embeddings provides a level of syntactic abstraction useful for dependency parsing, that level of abstraction is already explicit in constituency representations. Others that boost the performance when including word emnbeddings representations for syntactic parsing includes (Koo et al., 2008; Koo et al., 2010; Haffari et al., 2011; Tratz and Hovy, 2011).

Word embedding have also been applied to other (non-sequential NLP) tasks such as supersense tagging (Edouard Grave and Bach, 2013); grammar induction (Spitkovsky et al., 2011)

Do we want to address the following questions: How embeddings interact with the applications, for example, vocabulary expansion hypothesis (out of vocabulary words); static share hypothesis (in vocabulary words, e.g., individual first names are also rare in the treebank, but tend to cluster together in distributional representations); embedding structure hypothesis (e.g., group words by definiteness, like each, this, every, few, most, etc.).

Words embeddings are supposed to be useful for:

- connecting out of vocabulary words to known ones, - encouraging common behaviour among related in-vocabulary words, - directly providing features for the lexicon

3 Word Representations

Inspired by distributional semantics models, distributed word representation methods represent each word as a continuous vector, where similar words have a similar vector representation, therefore, capturing the similarity between words.

Example:

The resulting vectors can be used as features in many NLP applications and it has been shown that they outperform methods that treats words as atomic units ().

The estimation of the word vectors can be carry out with different models architectures. In this paper, we evaluate the word embedding learning methods presented in the next section and evaluate their utility in several sequence labelling tasks.

3.1 Word Embedding Learning Algorithms

- Glove (Pennington et al., 2014)
- Skip-gram (Mikolov et al., 2013)
- CBOW (Mikolov et al., 2013)
- Neural language model (**?**)
- Brown cluster (Brown et al., 1992)

3.2 Experimental Setup

3.3 Materials

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 3.3). The main reason of choosing these data set is that they are publicly available.

3.4 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.

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

Table 1: Corpus used to learn word embeddings

3.5 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, 400, 800.
- **Context window size**: 5, 10, 15.

Brown clustering:
Number of clusters : 250, 500, 1000, 2000, 4000

4 Sequence Tagging Tasks

We evaluate different learning approaches of word embeddings in four different sequence tagging tasks: POS tagging, chunking, MWE identification, and name entity recognition. Because we can either update pre-trained word embeddings during training or not, through the evaluation, we want to answer the following questions:

- How well do different word embeddings perform in all tasks when supervised fine-tuning is *not* performed?
- How well do different word embeddings perform in all tasks when supervised fine-tuning is performed?
- How does the size of labeled training data affect the experimental results?
- How well do the word embeddings perform for unknown words?

- How do the key parameters of each word learning algorithms affect the experimental results?

For each task, we feed learned embeddings into the graph transformer trained with sentence tag criterion (Turian et al., 2010). The graph transformer is equivalent to CRF, if we do not update word embeddings. For all tasks, we train Graph transformer with pre-trained word embeddings in the following two settings:

- CRF with conventional features.
- Graph transformer *does not* fine tune embeddings during training.
- Graph transformer fine tunes the embeddings during training.

For each task, we split the data into a training set, validation set, and a test set. The hyper parameters are tuned on the validation set with random search (Bergstra and Bengio, 2012). To be fair, for each model, we randomly choose 100 hyper parameter combinations and pick up the best one based on its performance on the validation set. Then each model is evaluated in a semi-supervised setting. We start with training models on 10% of the training data, and evaluate them on the test dataset. Then we incrementally add another 10% of the training data and evaluate them until all training data is used. We adopt per-word F1 scores as the evaluation metric for all tasks except POS tagging. We keep using per-word accuracy for POS tagging. In order to evaluate model performance on unknown words, we report also the average F1 scores for the words that do not occur in the training set.

In order to assess the influence of the key parameters of each word learning algorithms, we evaluate all embedding based models with varying key parameters on the full training set.

We also set up experiments to verify that CRF/graph transformer requires different feature design for different kinds of pre-trained word embeddings. One kind of word embeddings is represented by (Bengio et al., 2006). It maximises the word sequence likelihood with a model based on a weighted linear combination of word embedding features. Another kind of word embeddings is skip-gram and its variants, which is based on the dot product of two word embeddings. Therefore, we compare at least two ways of representing context word embedding features for each token: i) we concatenate word embeddings of context words within a fixed window as context features; ii) we concatenate the result of element-wise multiplication of current token embedding and each context word embedding as context features.

4.1 POS tagging

We could choose one of the options.

4.1.1 Option 1

Almost the same setting as (Collobert et al., 2011), except adding one more test set.
Training set: 0-18 of WSJ.
Validation set: 19-21 of WSJ.
Test set: 22-24 of WSJ, and English Web Treebank. We report model performances on these two test sets respectively.
Feature space: the same set as in (Collobert et al., 2011)

4.1.2 Option 2

Use the experimental setting in (Owoputi et al., 2013) for twitter POS tagging. All word embeddings will be learned from the twitter corpus provided by Scott.

4.2 Chunking

The same setting as (Turian et al., 2010)

Training set: WSJ train set.
Validation set: Randomly sampled 1000 sentences from the train set for development.
Test set: CoNLL2000 test set.
Feature space: the same set as in (Turian et al., 2010)

4.3 MWE Identification

Training set: randomly sampled 500 documents from Nathanas corpus.
Validation set: randomly sampled 100 documents from Nathanas corpus.
Test set: remaining 123 documents from Nathanas corpus..
Feature space: the same set as in (Schneider et al., 2014b)

Worth contacting Nathan as possible co-author?

4.4 Named entity recognition

Training set: CoNLL03 train set.
Validation set: CoNLL03 development set.

Test set: CoNLL03 test set and MUC7. We report model performances on these two test sets respectively.

Feature space: the same set as in (Turian et al., 2010)

5 Experimental Results and Discussion

Acknowledgments

Anonymised
Anonymised
Anonymised
Anonymised
Anonymised
Anonymised

References

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 822–827, Baltimore, USA.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815, Baltimore, USA.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305.

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.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Guillaume Obozinski Edouard Grave and Francis Bach. 2013. Hidden markov tree models for semantic class induction. In *Proceedings of CoNLL*.

Gholamreza Haffari, Marzieh Razavi, and Anoop Sarkar. 2011. An ensemble model that combines syntactic and semantic clustering for discriminative dependency parsing. In *ACL (Short Papers)*, pages 710–714. The Association for Computer Linguistics.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *In Proc. ACL/HLT*.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1288–1298, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Andriy Mnih and Geoffrey Hinton. 2008. A scalable hierarchical distributed language model. In *In NIPS*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *HLT-NAACL*, pages 380–390.

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

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014a. Discriminative lexical semantic segmentation with gaps: Running the mwe gamut. *Transactions of the Association of Computational Linguistics*, 2(1):193–206.

Nathan Schneider, Spencer Onuffer, Nora Kazour, Emily Danchik, Michael T. Mordowanec, Henrietta Conrad, and Noah A. Smith. 2014b. Comprehensive annotation of multiword expressions in a social web corpus. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 455–461, Reykjavík, Iceland, May. ELRA.

Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1281–1290, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1257–1268, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.