# Word Embeddings Evaluation and Combination

**Sahar Ghannay**[1], **Benoit Favre**[2], **Yannick Estève**[1] **and Nathalie Camelin**[1]
[1]LIUM - University of Le Mans, 72000, Le Mans, France
[2]Aix-Marseille Université, CNRS, LIF UMR 7279, 13000, Marseille, France

**Abstract**

Word embeddings have been successfully used in several natural language processing tasks (NLP) and speech processing. Different approaches have been introduced to calculate word embeddings through neural networks. In the literature, many studies focused on word embedding evaluation, but for our knowledge, there are still some gaps. This paper presents a study focusing on a rigorous comparison of the performances of different kinds of word embeddings. These performances are evaluated on different NLP and linguistic tasks, while all the word embeddings are estimated on the same training data using the same vocabulary, the same number of dimensions, and other similar characteristics. The evaluation results reported in this paper match those in the literature, since they point out that the improvements achieved by a word embedding in one task are not consistently observed across all tasks. For that reason, this paper investigates and evaluates approaches to combine word embeddings in order to take advantage of their complementarity, and to look for the effective word embeddings that can achieve good performances on all tasks. As a conclusion, this paper provides new perceptions of intrinsic qualities of the famous word embedding families, which can be different from the ones provided by works previously published in the scientific literature.

**Keywords:** Word embeddings, benchmarking, speech processing, natural language processing

## 1. Introduction

Word embeddings are projections in a continuous space of words supposed to preserve the semantic and syntactic similarities between them. They have been shown to be a great asset for several Natural Language Processing (NLP) tasks, like part-of-speech tagging, chunking, named entity recognition, semantic role labeling, syntactic parsing (Bansal et al., 2014a; Turian et al., 2010; Collobert et al., 2011), and also for speech processing: for instance, word embeddings were recently involved in spoken language understanding (Mesnil et al., 2015), in detection of errors in automatic transcriptions, and in calibration of confidence measures provided by an automatic speech recognition system (Ghannay et al., 2015).

These word representations were introduced through the construction of neural language models (Bengio et al., 2003; Schwenk, 2013). Different approaches have been proposed to compute them from large corpora. They include neural networks (Collobert et al., 2011; Mikolov et al., 2013a; Pennington et al., 2014), dimensionality reduction on the word co-occurrence matrix (Lebret and Collobert, 2013), and explicit representation in terms of the context in which words appear (Levy and Goldberg, 2014). One particular hypothesis behind word embeddings is that they are generic representations that shall suit most applications.

Many studies have focused on the evaluation of word embeddings intrinsic quality, as well as their impact when they are used as input of systems. Turian *et al.* (Turian et al., 2010) evaluate different types of word representations and their concatenation on the chunking and named entity recognition tasks.

This work was partially funded by the European Commission through the EUMSSI project, under the contract number 611057, in the framework of the FP7-ICT-2013-10 call, by the French National Research Agency (ANR) through the VERA project, under the contract number ANR-12-BS02-006-01, and by the Région Pays de la Loire.

The evaluation can be performed as well on the word similarity and analogical reasoning tasks, like in (Levy and Goldberg, 2014; Ji et al., 2015; Gao et al., 2014; Levy et al., 2015). Recently, the study proposed by (Levy et al., 2015), focuses on the evaluation of neural-network-inspired word embedding models (Skip-gram and GloVe) and traditional counted-based distributional models - pointwise mutual information (PMI) and Singular Value Decomposition (SVD) models-. This study reveals that the hyperparameter optimizations and certain system design choices have a considerable impact on the performance of word embeddings, rather than the embedding algorithms themselves. Moreover, it shows that, by adapting and transferring the hyperparameters into the traditional distributional models, they achieve similar gains as the neural-network word embeddings.

In this paper, we present a rigorous comparison of the performances of different kinds of word embeddings coming from different available implementations: word2vec (Mikolov et al., 2013a), GloVe (Pennington et al., 2014), CSLM (Schwenk, 2007; Schwenk, 2013) and word2vecf on dependency trees (Levy and Goldberg, 2014). Some of them were never compared; for instance, word2vec embeddings (Mikolov et al., 2013a) have been never compared to the CSLM toolkit, which is able to build deep feedforward neural network language models on large datasets because of an efficient code optimized for GPUs. Moreover, dependency-based word embeddings (Levy and Goldberg, 2014) have been never compared to CSLM, GloVe or Skip-gram (Mikolov et al., 2013a) embeddings.

In order to measure the supposed semantic and syntactic information captured by word embeddings, we evaluate their performance for different NLP tasks as well as on linguistic tasks.

In some state of the art studies (Mikolov et al., 2013a; Mikolov et al., 2013b; Bansal et al., 2014b), the evaluated word embeddings were estimated on different training data, or with different dimensionality. In this study all the word

embeddings are estimated on the same training data, using the same vocabulary, the same dimensionality, and the same window size.
In addition to these word embeddings evaluation, we are interested on their combination through concatenation, Principal Component Analysis and ordinary autoencoder in order to look for an effective embedding that can achieve good performance on all tasks.
The paper is organized along the following lines: section 2. presents the different types of word embeddings evaluated in this study. Section 3. describes the benchmark tasks. The experimental setup and results are described in section 4., and the conclusion in Section 5..

## 2. Word embeddings

Different approaches have been proposed to create word embeddings through neural networks. These approaches differ in the type of the architecture and the data used to train the model. In this study, we distinguish three categories of word embeddings: the ones estimated on unlabeled data based on simple or deep architectures, and others estimated from labeled data. These representations are detailed respectively in the next subsections.

### 2.1. Fast and simple estimation of word embeddings

This section presents three types of word embeddings coming from two available implementations:

- **CBOW**: This architecture, proposed by (Mikolov et al., 2013a), is similar to a feedforward Neural Network Language Model (NNLM) where the non-linear hidden layer is removed, and the contextual words are projected on the same position. It consists in predicting a word given its past and future context, by averaging the contextual word vectors and then running a log-linear classifier on the averaged vector to get the resultant word.

- **Skip-gram**: This second architecture from (Mikolov et al., 2013a) is similar to *CBOW*, trained using the negative-sampling procedure. It consists in predicting the contextual words given the current word. Also, the context is not limited to the immediate context, and training instances can be created by skipping a constant number of words in its context, for instance, $w_{i_{-3}}$,$w_{i_{-4}}$,$w_{i_{+3}}$,$w_{i_{+4}}$, hence the name *skip-gram*.

- **GloVe**: This approach is introduced by (Pennington et al., 2014), and relies on constructing a global co-occurrence matrix of words in the corpus. The embedding vectors are based on the analysis of co-occurrences of words in a window.

### 2.2. CSLM word embeddings

**CSLM** word embeddings are computed from unlabeled data by the CSLM toolkit (Schwenk, 2013), which estimates a feedforward neural language model. This approach projects the $n-1$ word indexes onto a continuous space and, from these word embeddings representations, computes the $n$-gram probabilities of each word in a short-list of the most frequent words as outputs of a the neural network. This architecture is more complex and more time-consuming to train than the three approaches presented above, but the computation time is reasonable due to the ability of the GPU implementations.

### 2.3. Dependency-based word embeddings

(Levy and Goldberg, 2014) proposed an extension of word2vec, called word2vecf and denoted **w2vf-deps**, which allows to replace linear bag-of-words contexts with arbitrary features. This model is a generalization of the skip-gram model with negative sampling introduced by (Mikolov et al., 2013a), and it needs labeled data for training. As in (Levy and Goldberg, 2014), we derive contexts from dependency trees: a word is used to predict its governor and dependents, jointly with their dependency labels. This effectively allows for variable-size.

## 3. Benchmark tasks

### 3.1. NLP tasks

In this sub-section, we briefly introduce the NLP tasks on which we evaluate the performance of the different word embeddings: part-of-speech tagging (POS), syntactic chunking (CHK), named entity recognition (NER), and mention detection (MENT).
For each of these tasks, a label has to be predicted for each word in context. Therefore we model the problem as feeding a neural network with the concatenation of the five word embeddings of a 5-gram as input. This 5-gram is centered on the word for which the prediction has to be made by the neural network. If an embedding does not exist for one of the words, it is replaced with 0. Words outside sentence boundaries are replaced with 0.
We test word embeddings in the context of the following tasks:

- Part-Of-Speech Tagging (POS): categorizing words among 48 morpho-syntactic labels (noun, verb, adjective, *etc.*). The system is evaluated on the standard Penn Treebank benchmark train/dev/test split (Marcus et al., 1993).

- Chunking (CHK): segmenting sentences in proto-syntactic constituents. There are 22 begin-inside-outside encoded word-level labels. The system is evaluated on the CoNLL 2000 benchmark (Tjong Kim Sang and Buchholz, 2000).

- Named Entity Recognition (NER): recognizing named entities in the text, such as persons, locations and organizations. There are 21 begin-inside-outside encoded word-level labels. The system is evaluated on the CoNLL 2003 benchmark (Tjong Kim Sang and De Meulder, 2003).

- Mention detection (MENT): recognizing mentions of entities for coreference resolution. There are 3 labels (begin, inside, outside). The task is performed on the Ontonotes corpus (Hovy et al., 2006) with the CoNLL 2012 split.

The description of the data split for each benchmark is summarized in table 1.

| Task | Benchmark | Train | Dev | Test |
|---|---|---|---|---|
| POS | Penn Treebank | 958K | 34K | 58K |
| CHK | CoNLL 2000 | 191K | 21K | 47K |
| NER | CoNLL 2003 | 205K | 52K | 47K |
| MENT | Ontonotes | 736K | 102K | 105K |

Table 1: Data split for each benchmark.

### 3.2. Linguistic tasks

In this study, we are interested as well on the analogical reasoning task for the purpose of testing the space sub-structures of the word embeddings. The tool provided by word2vec[1] and the Google analogy dataset (Mikolov et al., 2013a) are used for this task. The evaluation set is composed of five types of semantic questions such as capital cities (Athens:Greece $\rightarrow$ Tehran:?) and family (boy:girl $\rightarrow$ brother:?), and nine types of syntactic questions such as adjective-to-adverb (amazing:amazingly $\rightarrow$ calm:?) and comparative (bad:worse $\rightarrow$ big:?). Overall, there are 8,869 semantic and 10,675 syntactic questions. A question is correctly answered if the proposed word is exactly the same as the correct one. The question is answered using Mikolov (Mikolov et al., 2013a) approach named 3CosAdd (addition and subtruction) in the literature.
Finally, we want to evaluate the different word embeddings on a variety of word similarity tasks, based on corpora WordSim353 (Finkelstein et al., 2001), rare words (RW) (Luong et al., 2013) and, MEN (Bruni et al., 2012). These datasets contain word pairs with human similarity ratings. The evaluation of the word representations is performed by ranking the pairs according to their cosine similarities and measuring the Spearman's rank correlation coefficient with the human judgment.

## 4. Experiments

### 4.1. Experimental setup

The word embeddings described in section 2. are estimated on the annotated Gigaword corpus, which is composed of over 4 billion words. It contains dependency parses used for training w2vf-deps embeddings, and the unlabeled version is used to train the other embeddings. Note that words occurring less than 100 times have been discarded, resulting in a vocabulary size of 239K words. The parameter settings used in our experiments are summarized in Table2, their values have been selected based on previous studies (Levy and Goldberg, 2014; Ji et al., 2015; Gao et al., 2014; Levy et al., 2015).

| Embeddings | Win. | Dim. | Neg. |
|---|---|---|---|
| CBOW | 5 | 200 | 5 |
| Skip-gram | 5 | 200 | 5 |
| GloVe | 5 | 200 | - |
| CSLM | 5 | 200 | - |
| w2vf-deps | - | 200 | 5 |

Table 2: Parameters used for extracting the word embeddings (window size, dimension, negative sampling)

[1]https://code.google.com/p/word2vec/

The 5-gram NNLM used to compute the CSLM word embeddings is composed of a projection layer of 800 units, corresponding to 200-dimensional word embeddings, two hidden layers of 1024 units each, and an output layer providing probabilities for a short-list composed of the 16,384 most frequent words. The CSLM training process needed 16 hours and 30 minutes on a computer equipped with a NVIDIA Tesla K40 GPU card, while 8h was necessary for GloVe embeddings, 7h for Skip-gram, and about 3 hours and 30 minutes for CBOW.

### 4.2. Experimental results of individual word embeddings

#### 4.2.1. NLP tasks

In this section, we report the performance of the different word embeddings on the four NLP tasks. A neural network classifier based on a multi-stream strategy is used to train the models. This architecture depicted in figure 1, was introduced by (Ghannay et al., 2015) for the ASR error detection task.

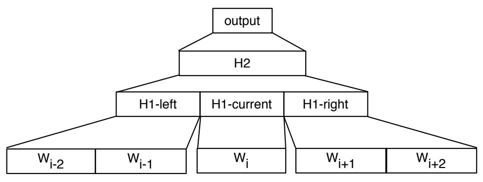

Figure 1: Architecture of the NN used for experiment on NLP tasks

The first hidden layer has 300, 100 and 300 units for H1-left, H1-current and H1-right respectively. The second hidden layer has 300 units. Activation functions are rectified linear units (relu) for the first layer and *tanh* for the second one. The hyper-parameters, learning rate and batch size, are tuned over the validation set available for each task. The CHK, NER and MENT tasks are evaluated by computing F1 scores over segments produced by our models. The POS task is evaluated by computing per-word accuracy. The *conlleval* script is used for evaluation (Mesnil, 2015). Last, the significance of our results is measured using the 95% confidence interval.

| Task | POS | CHK | NER | MENT |
|---|---|---|---|---|
| **Embeddings** | **Acc.** | **F1** | **F1** | **F1** |
| CBOW | 96.01 | 90.48 | 78.32 | 55.49 |
| Skip-gram | 96.43 | 89.64 | 77.65 | 57.80 |
| GloVe | 95.79 | 86.90 | 76.45 | 54.49 |
| CSLM | 96.24 | 90.11 | 76.20 | 57.34 |
| w2vf-deps | **96.66** | **92.02** | **79.37** | **58.06** |

Table 3: Performance of word embeddings on the NLP tasks.

Experimental results are summarized in Table 3. We observe that w2vf-deps embeddings reach the highest score for all tasks. This performance is related to the use of dependency based syntactic contexts, which capture different information more than the bag-of-word contexts. Nevertheless, the estimation of this embeddings require labeled

data, which can be difficult to provide for resource-scarce languages which do not have dependency parsers.
Considering the simple embeddings, we observe that Skip-grams performs significantly better than CBOW and GloVe on POS and MENT tasks. However, for the other tasks CBOW achieves the best results. Lastly, these embeddings outperforms CSLM for all tasks.

#### 4.2.2. Linguistic task 1: Analogical reasoning tasks

| **Subtask** | **CBOW** | **Skip-gram** | **GloVe** | **CSLM** | **w2vf-deps** |
|---|---|---|---|---|---|
| Capital cities | 89.5 | 88.3 | 93.1 | 34.0 | 71.5 |
| Capital-world | 81.0 | 88.2 | 92.2 | 16.1 | 34.4 |
| Currency | 9.5 | 17.6 | 16.6 | 1.1 | 8.8 |
| City-in-state | 22.9 | 27.2 | 36.2 | 3.5 | 6.2 |
| Family | 86.8 | 76.5 | 81.4 | 66.8 | 74.3 |
| Adjective-to-adverb | 13.3 | 18.2 | 22.3 | 7.3 | 5.3 |
| Opposite | 24.9 | 34.1 | 22.5 | 20.4 | 36.9 |
| Comparative | 81.4 | 79.3 | 84.8 | 70.1 | 87.6 |
| Superlative | 61.2 | 69.4 | 65.0 | 45.0 | 71.5 |
| Present-participle | 62.6 | 65.3 | 66.7 | 39.9 | 60.1 |
| Nationality-adjective | 81.1 | 86.7 | 91.2 | 25.6 | 25.8 |
| Past-tense | 55.1 | 56.7 | 59.2 | 54.1 | 55.9 |
| Plural | 54.0 | 55.0 | 69.8 | 17.0 | 59.9 |
| Plural-verbs | 37.8 | 61.8 | 48.4 | 48.5 | 86.8 |
| Semantic Acc. | 58.8 | 63.7 | **68.8** | 15.1 | 28.5 |
| Syntactic Acc. | 53.2 | 57.5 | **58.7** | 36.4 | 54.3 |
| Overall Acc. | 57.2 | 62.3 | **65.5** | 27.4 | 42.70 |

Table 4: % Accuracy of various word embeddings on the evaluation set of analogical reasoning tasks.

We observe in table 4 that the different word embeddings yield a large range of accuracy on this task. The word embeddings ranking obtained in the previous evaluation task is not preserved. Globally, GloVe achieves the best accuracy, followed by Skip-gram and CBOW embeddings. They achieve 65.5%, 62.3% and 57.2% of accuracy respectively. Thus, this result match those presented by (Pennington et al., 2014; Levy et al., 2015). While w2vf-deps and CSLM have respectively 43.1% and 27.4% of accuracy.

#### 4.2.3. Linguistic task 2: Similarity task

| **Task** | **WS353** | **RW** | **MEN** |
|---|---|---|---|
| CBOW | **59.0** | 46.5 | 60.9 |
| Skip-gram | 55.8 | **50.2** | **66.2** |
| GloVe | 53.3 | 41.0 | 66.0 |
| CSLM | 47.8 | 43.4 | 48.2 |
| w2vf-deps | 52.3 | 43.5 | 55.7 |

Table 5: Performance of word embeddings on word similarity tasks.

Table 5 summarizes the performance of word embeddings on similarity tasks. As we can see, the results are in favor of Skip-grams. In fact, it reaches the best results in two tasks, and based on confidence interval evaluation, it achieves nearly the same results as CBOW in WS353 task.

### 4.3. Performance of combined word embeddings

The evaluation of the different word embeddings reported in section 4.2., shows that the best embbedings are w2vf-deps, Skip-gram and GloVe. Each of them is efficient on one task. However, building an effective word embedding remains an ultimate goal, which can be achieved by the combination of embeddings.
Based on state of-the-art studies, the combination of different word embeddings takes advantage of their complementarity and yields an improvement on different tasks: chunking, and named entity recognition as in (Turian et al., 2010). For instance, as shown above , the simple concatenation of Brown clusters and word embeddings resulted in an improvement on chunking and named entity recognition. Moreover, in (Ghannay et al., 2015), we have investigated the use of different approaches to combine 100-dimensional word embeddings: concatenation (Concat), PCA and auto-encoders (AutoE). In that work, we have shown that the combination with auto-encoders yields significant improvement on the ASR error detection task.
Here, we propose to combine the simple word embeddings (CBOW, Skip-gram and GloVe) and the ones achieving the best results reported in section 4.2. (w2vf-deps, Skip-gram and GloVe), using the same approaches as in (Ghannay et al., 2015). The two combination sets are called *Simple* and *Best* respectively in the remainder of the paper.
The combination approaches are briefly detailed as follow:

**Concat:** For the first approach, we simply use the concatenation of the three word embeddings types from each combination set. As a consequence, each word is represented by a 600-dimensional vector.

**PCA:** For the second approach, the PCA technique is applied to Concat embeddings. According to these embeddings, the matrix composed of all words is first mean centering using Z-scores. The new coordinate system is then obtained computing PCA using the correlation method. The data is then projected onto the new basis considering only the first 200 components.

**AutoE:** Lastly, we investigate the use of ordinary auto-encoder (Vincent et al., 2008). This auto-encoder is composed of one hidden layer with 200 hidden units each. It takes as input the Concat embeddings and as output a vector of 600 nodes. For each word, the vector of numerical values produced by the hidden layer will be used as the combined word embedding.

The performance of the combined word embeddings are compared to the individual embeddings that it contains. Furthermore, the autoencoder is tuned on the dev corpus of NER task. In the following sections, the improvements are indicated in bold, whereas, based on confidence interval evaluation the significant ones are underlined.

#### 4.3.1. NLP tasks

As shown in table 6, the combination of word embeddings is helpful and yields significant improvement in CHK, NER and MENT tasks in almost all cases. For the POS task, the Best-Concat and Best-AutoE combined embeddings results are nearly the same as the best individual ones. Moreover, they achieve the best results on most NLP tasks.

| Task | | POS | CHK | NER | MENT |
|---|---|---|---|---|---|
| Dim. | Embeddings | Acc. | F1 | F1 | F1 |
| Simple: Cbow-deps-Skip-GloVe | | | | | |
| 600 | Simple-Concat | 96.24 | **91.24** | **79.43** | **57.86** |
| 200 | Simple-PCA | 96.39 | 90.20 | **78.99** | 57.72 |
| | Simple-AutoE | 95.99 | 89.59 | **78.44** | 57.76 |
| Best: w2vf-deps-Skip-GloVe | | | | | |
| 600 | Best-Concat | **96.67** | 91.88 | **81.06** | **58.20** |
| 200 | Best-PCA | 96.45 | 90.13 | **79.66** | **60.22** |
| | Best-AutoE | 96.64 | 91.35 | **80.43** | **60.39** |

Table 6: Performance of combined word embeddings on the NLP tasks.

#### 4.3.2. Linguistic task 1: Analogical reasoning task

As shown in table 7, the significant improvements for this task are achieved by the combination of the best embeddings, through the concatenation and PCA. These embeddings achieve respectively 71.4% and 70.7% of overall accuracy.
However, this is not the case for the Autoencoder combined word embeddings, which, achieve the lowest accuracy.

| Dim. | 600 | 200 | | 600 | 200 | |
|---|---|---|---|---|---|---|
| Subtask | Simple-Concat | Simple-PCA | Simple-AutoE | Best-Concat | Best-PCA | Best-AutoE |
| Capital cities | 92.9 | 92.7 | 90.1 | 94.7 | 94.7 | 90.3 |
| Capital-world | 89.0 | 89.6 | 82.4 | 93.8 | 95.3 | 85.5 |
| Currency | 14.1 | 14.4 | 9.4 | 23.5 | 23.9 | 18.3 |
| City-in-state | 30.6 | 34.9 | 33.6 | 40.0 | 39.9 | 24.6 |
| Family | 89.3 | 81.2 | 82.6 | 88.9 | 85.4 | 87.9 |
| Adj-to-adv | 17.4 | 12.3 | 11.7 | 25.2 | 20.9 | 14.5 |
| Opposite | 28.3 | 24.8 | 18.7 | 37.9 | 29.7 | 24.3 |
| Comparative | 84.6 | 81.6 | 80.6 | 91.2 | 90.0 | 83.6 |
| Superlative | 66.9 | 66.2 | 52.0 | 81.6 | 78.4 | 60.3 |
| Present-participle | 66.4 | 62.9 | 60.7 | 79.1 | 78.1 | 64.3 |
| Nationality-adj | 84.1 | 86.8 | 82.8 | 88.7 | 89.1 | 82.4 |
| Past-tense | 58.1 | 59.9 | 50.1 | 69.6 | 63.7 | 59.9 |
| Plural | 64.7 | 61.9 | 51.1 | 76.5 | 69.1 | 67.0 |
| Plural-verbs | 41.6 | 43.9 | 36.4 | 80.3 | 79.9 | 71.9 |
| Semantic Act | 65.8 | 66.9 | 59.6 | **70.0** | **72.5** | 62.5 |
| Syntactic Act | 57.4 | 55.4 | 50.2 | **72.6** | **69.2** | 61.7 |
| Overall | 62.9 | 62.8 | 56.0 | **71.4** | **70.7** | 62.0 |

Table 7: % Accuracy of various combined word embeddings on the evaluation set of analogical reasoning tasks.

#### 4.3.3. Linguistic task 2: Similarity task

Results on this task, as shown in table 8, are again in favor of the combination of the best embeddings. The concatenation and PCA combined embeddings yield results as good as the individual embeddings they contain on both WS353 and MEN tasks. However, among the combinations of the simple ones, the concatenation and PCA combined embeddings achieve an improvement respectively on WS353 and MEN tasks.
As in the analogical reasoning task, autoencoders result in lower performance. We have yet to find a definitive explanation to this behavior, but one conjecture is that the combination through autoencoders do not preserve the linear structure of the embeddings which allow translations to represent linguistic and semantic properties.

| Dim. | Task | WS353 | RW | MEN |
|---|---|---|---|---|
| Simple: Cbow-Skip-GloVe | | | | |
| 600 | Simple-Concat | **60.2** | 48.0 | 65.0 |
| 200 | Simple-PCA | 57.4 | 49.6 | **66.9** |
| | Simple-AutoE | 55.3 | 46.0 | 63.9 |
| Best: w2vf-deps-Skip-GloVe | | | | |
| 600 | Best-Concat | **57.0** | 48.6 | **69.4** |
| 200 | Best-PCA | **57.9** | 49.5 | **71.3** |
| | Best-AutoE | 55.8 | 44.6 | 64.9 |

Table 8: Performance of combined word embeddings on word similarity tasks.

## 5. Conclusions

In this paper, we perform a systematic comparison of major word embeddings impact on typical NLP tasks, as well as semantic and syntactic similarity tasks.
The evaluation results reported in this paper match those in the literature, since improvements achieved by one word embedding in a specific task are not observed in other tasks. We have confirmed that embeddings trained given dependency parses give the best performance on the NLP tasks. Thus, it is interesting to evaluate the performance of such embedding on the ASR error detection task. For the linguistic tasks, the results are in favor of the basic embeddings especially Skip-gram and GloVe. More, the basic embeddings outperform CSLM on all tasks.
Furthermore, we have proven, that the combination of the embeddings yields significant improvement. This result corroborates a previous observation made in recent work on embeddings combination for ASR error detection (Ghannay et al., 2015).
In addition, results obtained by Best-PCA show that building an effective word embedding that achieve good performance in almost all tasks, can be reached by the combination of the efficient embeddings in each task through PCA.
Finally, such combination performs poorly on intrinsic analogical reasoning tasks. This peculiar aspect, which seems to indicate that NLP systems do not make use of semantic regularities presented in embeddings, remains to be explored in future work.

## 6. References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014a. 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. Association for Computational Linguistics.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014b. Tailoring continuous word representations for dependency parsing. In *ACL (2)*, pages 809–815.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. volume 3, pages 1137–1155. JMLR.org, March.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145, Jeju Island, Korea, July. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. volume 12, pages 2493–2537. JMLR.org.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Bin Gao, Jiang Bian, and Tie-Yan Liu. 2014. Wordrep: A benchmark for research on learning word representations. *CoRR*, abs/1407.1640.

Sahar Ghannay, Yannick Estève, Nathalie Camelin, Camille Dutrey, Fabian Santiago, and Martine Adda-Decker. 2015. Combining continous word representation and prosodic features for asr error prediction. In *3rd International Conference on Statistical Language and Speech Processing (SLSP 2015)*, Budapest (Hungary), November 24-26.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Shihao Ji, Hyokun Yun, Pinar Yanardag, Shin Matsushima, and S. V. N. Vishwanathan. 2015. Wordrank: Learning word embeddings via robust ranking. *CoRR*, abs/1506.02761.

Rémi Lebret and Ronan Collobert. 2013. Word emdeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*.

Omer Levy and Yoav Goldberg. 2014. Dependencybased word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(3):530–539.

Grégoire Mesnil. 2015. Recurrent Neural Networks with Word Embeddings DeepLearning 0.1 documentation. http://www.deeplearning.net/tutorial/rnnslu.html#rnnslu.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, volume 12.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Holger Schwenk. 2013. CSLM-a modular open-source continuous space language modeling toolkit. In *INTERSPEECH*, pages 1198–1202.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. 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.

P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*.