Multi-Sense Embeddings Through a Word Sense Disambiguation Process

Terry Ruas
CIS Department
University of Michigan
Dearborn, MI, USA
truas@umich.edu

William Grosky
CIS Department
University of Michigan
Dearborn, MI, USA
wgrosky@umich.edu

Akiko Aizawa
University of Tokyo
National Institute of Informatics
Tokyo, Japan
aizawa@nii.ac.jp

Abstract

Natural Language Understanding has seen an increasing number of publications in the last years, especially after robust word embedding models became popular. These models gained a special place in the spotlight when they proved themselves able to capture and represent semantic relations underneath huge amounts of data. Nevertheless, traditional models often fall short in intrinsic issues of linguistics, such as polysemy and homonymy. Multi-sense word embeddings were devised to alleviate these and other problems by representing each word-sense separately, but studies in this area are still in its infancy and much can be explored. We follow this scenario by proposing an unsupervised technique that disambiguates and annotates words by their specific sense, considering their context influence. These are later used to train a word embeddings model to produce a more accurate vector representation. We test our approach in 6 different benchmarks for the word similarity task, showing that our approach can sustain good results and often outperforms current state-of-the-art systems.

1 Introduction

Semantic analysis is probably one of the oldest challenges in Natural Language Processing (NLP) and still present in almost all of its downstream applications. The translation of semantics to a computer is not an easy task to accomplish. Even among humans, the concept of *what is semantics* is not unanimous, which leads to multiple interpretations and make things even more challenging (Putnam, 1970).

In spite of being a classical problem, its popularity continues to draw attention of many research projects. Grosky and Ruas (2017) analyzed 2,872 scientific publications (e.g. papers, journals, reports) between 2005 and 2015, revealing an increasing trend in publications involving *Semantics* and *Context Aspects* in different areas of multimedia. In these publications, methods applying different techniques try to capture semantic characteristics of text documents using clever approaches, such as: latent semantic analysis, word embeddings, machine learning, artificial neural networks (NN) and others.

After recent contributions (Bengio et al., 2003; Mikolov et al., 2013a; Mikolov et al., 2013b; Pennington et al., 2014), word embeddings techniques have received much attention in the NLP community. They represent words or phrases by real number vectors which are used to extract relationships between them. Their overall performance have demonstrated superior results in many different NLP tasks, such as: chunking (Dhillon et al., 2011), meaning representation (Bordes et al., 2012), machine translation (Mikolov et al., 2013b), relation similarity (RS) (Mikolov et al., 2013c; Iacobacci et al., 2015), sentiment analysis (Socher et al., 2013), word sense disambiguation (WSD) (Navigli, 2009; Chen et al., 2014), word similarity (WS) (Neelakantan et al., 2014; Chen et al., 2015; Camacho-Collados et al., 2015; Iacobacci et al., 2015) and topic categorization (Pilehvar et al., 2017).

Notwithstanding their robustness, most conservative word embedding approaches fail to deal with polysemy and homonymy problems (Li and Jurafsky, 2015). Recently, researchers have been trying to improve their semantic representations by producing multiple vectors (multi-sense embeddings) based on the word's sense, context and distribution in the corpus (Reisinger and Mooney, 2010b; Huang et al., 2012). Another concern with traditional techniques is that they often neglect exploring lexical structures with valuable semantic relations, such as: WordNet (WN) (Fellbaum, 1998), ConceptNet (CN) (Liu and Singh, 2004) and BabelNet (BN) (Navigli and Ponzetto, 2012). Some publications take advantage of these structures and combine them to multi-sense representations, improving their overall performance even more (Camacho-Collados et al., 2015; Rothe and Schütze, 2015; Iacobacci et al., 2015; Li and Jurafsky, 2015; Iacobacci et al., 2016; Pilehvar and Collier, 2016; Mancini et al., 2017).

In this paper, we propose a model that allows us to obtain specific word-sense vectors from any non-annotated text document as input. For this, we extend the disambiguation algorithm presented by Ruas and Grosky (2017) to find the most suitable sense of a word based on its context, which is later trained into a NN to produce specific vectors. The benefits of our approach are fivefold. First, we provide an unsupervised annotation algorithm that takes the context of each word into consideration. Second, our model disambiguates words from any part-of-speech (POS), mitigating issues with polysemy and homonymy. Third, the annotation and training steps in our approach are independent, so if more robust algorithms are available they can be easily incorporated. Fourth, the generated vectors keep the same algebraic properties as traditional word embedding models, such as: *vec(king) - vec(man) + vec(woman)* $\approx$ *vec(queen)*. Lastly, the produced vectors can be used in the system iteratively to improve the model's performance in the disambiguation and training parts. To validate the quality of our work, we test our approach in 6 different benchmarks for WS, showing that our model can sustain good results and sometimes outperform current state-of-the-art systems.

2 Related Work

Distributed representation of words from documents has received substantial attention from the NLP community in the last years, especially after the extremely popular *word2vec* proposed by Mikolov et al., (2013a; 2013b). However, the idea of representing words in an n-dimensional space goes back to the 20th century with the bag-of-words (BOW) (Harris, 1954). Despite its simplistic methodology, the BOW approach brings many disadvantages, such as: word order is lost, data sparsity and high dimensionality, to name a few. Bengio et al., (2003) try to solve the latter problem by proposing a neural probabilistic language model that learns a representation while keeping a compact probability distribution of word sequences. Collobert and Weston (2008) later defined a faster general single convolutional network architecture showing that multitask and semi-supervised learning can improve the generalization in shared tasks, such as: POS tagging, morphological segmentation, named entity recognition (NER) and WS. Beside these, other language prediction tasks are also popular in the NLP community (Schwenk et al., 2006; Zhong and Ng, 2010; Turian et al., 2010; Turney and Pantel, 2010; Bordes et al., 2012; Zou et al., 2013)

It is undeniable that word2vec's contributions with continuous skip-gram (SG) and continuous bag-of-words (CBOW) from Mikolov et. al., (2013a; 2013b) brought a legion of new publications to the NLP, or more specifically, the word embeddings, arena. Its popularity is due to, among other things, the efficient log-linear NN language model, robustness and low dimensionality vector representation. Both approaches make vector representations of words with similar contexts to have similar values, a theory described in the *Distributional Hypothesis* which states "*a word is characterized by the company it keeps*" (Firth, 1957). In the CBOW training model, one tries to predict a word given its neighboring context, while SG does the inverse, predicting the context given a target word. Additional word embedding representation are also explored by GloVe (Pennington et al., 2014) and SENNA (Collobert et al., 2011).

(8,38) Even though our approach has a disambiguation step in its architecture, the presented experiments and discussions focus on how the combination of our WSD and word embeddings can be used to improve results in the WS task and benefit one another. We do have future plans to compare our WSD technique with alternative methods, but for now this is beyond the scope of this paper. A considerable number of publications and panorama of the field is detailed by Navigli's (2009) survey.

(27)Nearly all publications in embedding words into single vectors often suffer from the same problem, ambiguous words are represented in a unique vector. In other words, polysemy and homonymy are not handled properly. For example, in the sentence "This *club* is great!" is not clear if the term *club* is related to the sense of *baseball club*, *clubhouse*, *golf club* or any other. Systems that use standard word embeddings, like word2vec or GloVe, will most likely represent all possible meanings for the term *club* in one single n-dimensional vector.

Some researchers try to solve this representation limitation by producing separate vectors for each word-sense. Even though the number of publications in this area is still small, their early findings demonstrated encouraging results in many NLP challenges (Li and Jurafsky, 2015). One of the earliest models was proposed by Reisinger and Mooney (2010b), and Huang et al., (2012). Both of them work with the concept of clustering word-senses by their context. The former follows a probabilistic approach to produce a multi-prototype vector space model, using word sense discovery to evaluate a word's context. They set K clusters to represent the different contexts where the word is used. The latter introduces a NN language model capable of distinguishing the semantics of words by considering their global and local clusters representing their context. Trask et al., (2015) extended Huang et al., (2012)'s model by leveraging supervised NLP labels, instead of relying on unsupervised clusters techniques to produce specific word-sense vectors.

Other techniques also take advantage of probabilistic models to learn their own representation for each sense. Tian et al., (2014) design an efficient expectation maximization algorithm integrated with the SG model to avoid the issues brought by clustering based approaches. Another modification of SG is proposed by Neelakantan et al., (2014), in which they introduce the *Multi-Sense Skip-Gram* (MSSG) model. Their technique performs word sense discrimination and embedding at the same time, improving its efficiency. In the MSSG version, they assume a specific number of senses for each word, while in the *Non-Parametric Multi-Sense Skip-Gram* (NP-MSSG) this number varies. Nieto Piña and Johansson (2015) also explore the SG training phase by initializing all possible variations of word-senses, but only train the most probable ones according to their maximization objective. Other publications using pre-trained word embeddings to process them into word-sense vectors are described in (Pilehvar and Collier, 2016; Johansson and Piña, 2015).

In multi-sense embeddings approaches the use of lexical resources to improve their performance in NLP tasks is quite common. WN[1], CN[2] and BN[3] are examples of popular resources used to obtain word-sense vectors. Based on BN, Iacobacci et al.'s, (2015) system learns word-sense embeddings for WS and RS tasks, moving from a word to sense embedding representation. Rothe and Schütze (2015; 2017) use WN in their so called *AutoExtend* to produce token embeddings from a set of synonyms (*synsets*) and lexemes using a pre-existing word embeddings model. Their approach is detached from any word-type representation, so it can be easily translated to other learning techniques. Camacho-Collados et al., (2015; 2016) proposed a semantic vector representation for BN called *NASARI* which is extended to a multilingual scenario. Iacobacci et al., (2016) showed how WSD systems can have their performance improved if word embeddings are used. Pilehvar et al., (2017) developed a graph-based WSD algorithm to improve performance in downstream NLP applications (e.g. topic categorization, polarity detection). More recently, Mancini et al.,'s (2017) algorithm associates the words to the most connected senses in a sentence to produce their embeddings.

Chen et al., (2014) system performs WSD on vector embeddings and uses them to learn word-sense representations from the relevant occurrences. Their WSD technique considers only the most similar words in WN's *glosses*[4] in the process. Likewise, Chen et al., (2015) also use WN's glosses and context clustering to produce word-sense vectors via a convolutional NN, which also needs to be trained.

[1]`https://wordnet.princeton.edu`
[2]`http://conceptnet.io`
[3]`https://babelnet.org`
[4]`https://wordnet.princeton.edu/documentation/wngloss7wn`

Our approach, on the other hand, considers all the words in WN's glosses, not just the most similar ones. In addition, there is no extra training or hyperparameter adjustments other than those required by a standard word2vec implementation.

3 (3, 12) Synset Disambiguation, Annotation and Embedding

The main idea of our process is to have a modular system with two independent tasks: (i) disambiguation followed by annotation and (ii) word embeddings training. This configuration allows us to incorporate more robust techniques in the future, specially for the training step. In the first task, we process a collection of articles (documents) from two Wikipedia Dumps (Section 5) to transform each word in the corpus into a synset by using WN as our lexical resource (Miller, 1995; Fellbaum, 1998). This is done through one of two algorithms: *Most Suitable Sense Annotation (MSSA)* (Section 3.1), *Most Suitable Sense Annotation N Refined (MSSA-NR)* (Section 3.2) and *Most Suitable Sense Annotation - Dijkstra (MSSA-D)* (Section 3.3). In the second task, we use Mikolov et al., (2013a; 2013b)'s word2vec to train the produced corpus and obtain n-dimensional vectors of each word-sense, represented by synsets (multi-sense embeddings).

(3) In their initial form, both MSSA and MSSA-D use Google News vectors[5] to help disambiguate the word-senses in the corpus. MSSA works locally, trying to choose the best representation for a word-sense given its context window, one at a time. MSSA-D on the other hand, has a more global perspective since it considers the lowest overall cost of the word-senses from first to the last word in a document. Once the synset embeddings models are available, we can feed the system again and improve the disambiguation step in either MSSA or MSSA-D algorithms, relieving it from the original Google News vectors dependency. We call this approach MSSA-NR, where N represents the number of iterations for the produced synsets is used in the disambiguation setp. Different from other systems (Chen et al., 2014; Chen et al., 2015; Rothe and Schütze, 2015), our method has only one training phase and does not rely on any extra hyperparameters other than those required in the original word2vec implementation. In the following sections we will explain the details of our approach illustrated in Figure 3.

[5]`https://code.google.com/archive/p/word2vec/`

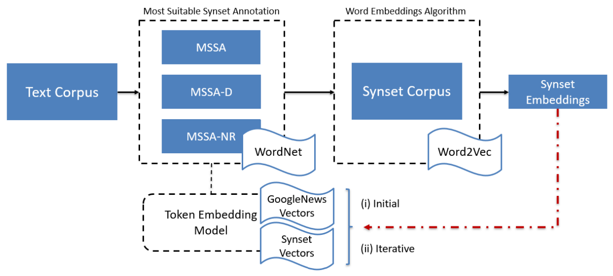

Figure 1: General system architecture of MSSA, MSSA-D and MSSA-NR.

3.1 (3) Most Suitable Sense Annotation (MSSA)

As Ruas and Grosky (2017) presented, each evaluated word w_i takes into consideration its context, represented by its surrounding neighboring words, w_{i-1} and w_{i+1} as illustrated in Algorithm 1.(38) We also use WN as our lexical database to extract all synsets from each word in the text, but our algorithm works for any word mapped in WN, not just nouns. (39) In our approach, all text is preprocessed by: normalizing all tokens in lowercase, removing punctuation, html tags, numbers, common English stopwords and discarding all words not present in WN. After the initial data cleaning, we extract all pair of synsets and glosses for each word w_i in a sliding context window of 3 words. (lines 4:13). (18) Our hypothesis is similar to the one proposed by Mikolov et al., (2013a)' CBOW, which uses the context to predict a given word. However, since our algorithm considers all synsets from w_i, w_{i-1} and w_{i+1}, we currently limit this word context window to restrict the number of comparisons necessary to infer the most suitable meaning for w_i. It is in our plans to incorporate a larger context without compromising the overall performance for this step. Next, after removing common English words from the glosses, (19) we retrieve and average the embeddings from the remaining tokens on it by using Google News vectors[6], which we call $gloss - average - vector$. (11) If there are no remaining tokens in the gloss or no vectors in the model an empty vector will be assigned for that synset-gloss pair. However, this scenario is very unlikely since the words in the glosses have their vector extracted from a model trained on a huge corpus. This process is done for all synset-glosses for each element w_i ($current$), w_{i-1} ($former$) and w_{i+1} ($later$) respectively, (29) where N, M and P represent their total number of available synset-glosses per word (lines 14:17). The gloss-average-vector for each synset, from the words in the sliding window, is used to calculate the similarity with its immediate neighbors and the sense for $a\_current$ with the highest value is stored. In other words, for each word w_i we calculate the cosine similarity against w_{i-1} and w_{i+1} with respect to their synset-glosses vectors and return the synset for $a\_current$ that produces the best result (line 18). The synset with the highest similarity, regarding its neighbors word-sense, is then chosen to represent each word w_i and added to a list of tokens (Line 19). The first and the last tokens are treated differently since they do not have a complete context window (lines 8 and 11).

In the initial configuration, we use Google News vectors as our standard word embeddings model (tm in Lines 14:17), which was trained over 100 billion words and contains 300-dimensional vectors for 3 million unique words and phrases (Mikolov et al., 2013b). This model can work as an initialization seed in our approach, as we can use the synsets embeddings and re-feed our system so in the next iteration we use our calculated vectors of synsets to disambiguate the word-senses in the corpus.

Algorithm 1 Most Suitable Sense Annotation (MSSA)

```
Require: $d = \{w_i, ..., w_n\} : w_i\exists$ in lexical database (WordNet)
 1: function MSSA($d$, $tm$, $ld$)                          ▷ Where
    $d$ - document containing words $w_n$, $tm$ - trained word embedding
    model, $ld$ - lexical data base
 2:     list_of_tokens = $\emptyset$
 3:     for $i = 0$ to $n$ do
 4:         $current$ = synset-glosses($w_i$, $ld$)
 5:         if $i \neq 0 \wedge i \neq n$ then
 6:             $former$ = synset-glosses($w_{i-1}$, $ld$)
 7:             $latter$ = synset-glosses($w_{i+1}$, $ld$)
 8:         else if $i = 0$ then
 9:             $former$ = None
10:             $latter$ = synset-glosses($w_{i+1}$, $ld$)
11:         else
12:             $former$ = synset-glosses($w_{i-1}$, $ld$)
13:             $latter$ = None
14:         for $s_c$,$s_f$,$s_l$ in ($current, former, latter$) do     ▷ Where
            $0 \leq c \leq N, 0 \leq f \leq M$ and $0 \leq s \leq P$
15:             $a\_current \longleftarrow$ gloss-avg-vec($s_c$, $tm$)
16:             $a\_former \longleftarrow$ gloss-avg-vec($s_f$, $tm$)
17:             $a\_latter \longleftarrow$ gloss-avg-vec($s_l$, $tm$)
18:         $t$ = best_score($a\_current$, $a\_former$, $a\_latter$)
                ▷ (30) Where the cosine similarity is performed between
            ($a\_current$,$a\_former$) and ($a\_current$,$a\_latter$), returning the
            $synset$ with the highest score for $a\_current$
19:         list_of_tokens $\longleftarrow t$
20:     return $list\_of\_tokens(synsets)$
```

3.2 (20,21,3)Most Suitable Sense Annotation N Refined (MSSA-NR)

Once we train our own model based on synset tokens, we can feed the system with it again and obtain a faster synset representation to be trained in the word embedding algorithm. We hypothesize that by using a disambiguated and granular embeddings we will obtain a more refined synset model. The algorithm for this approach is similar to the one presented in Section 3.1, so we are still using the same training cleaned corpus composed by Wikipedia articles, but the processing inner steps are a little different.

We identify this refined case as *MSSA-NR*, where N represents the number of times we feed the system with the generated vectors by our own approach. Algorithm 2 starts similar to Algorithm 1 as we also use WN as our lexical database and still work with the same sliding context window for the words. The main difference relies between lines 4 and 17, where since our embeddings are made of synsets we do not need to extract the pairs of synset-glosses nor calculate the gloss-average-vector for each synset. Instead, we just extract all synsets available in WN for w_i ($current$), w_{i-1} ($former$) and w_{i+1} ($later$) (lines 4:13) and directly retrieve their respective vector embeddings from the synset model trained, where Q, R and S represent their total number of available synsets per word (lines 4:13). Since MSSA-NR is using an embedding model on the same corpus it was firstly generated, all the words will have at least one synset mapped, so there is no risk of not finding a vector for a given word. After we retrieve the vec-

[6]https://code.google.com/archive/p/word2vec/

Algorithm 2 Most Suitable Sense Annotation N Refined (MSSA-NR)

```
Require: $d = \{w_i, ..., w_n\}$ : $w_i \exists$ in lexical database (WordNet)
 1: function MSSA-NR($d$, $tsm$, $ld$)    ▷ Where $d$ - document containing words $w_n$, $tsm$ - trained synset embedding model, $ld$ - lexical data base
 2:     list_of_tokens = $\emptyset$
 3:     for $i = 0$ to $n$ do
 4:         $current$ = synsets($w_i$, $ld$)
 5:         if $i \neq 0 \wedge i \neq n$ then
 6:             $former$ = synsets($w_{i-1}$, $ld$)
 7:             $latter$ = synsets($w_{i+1}$, $ld$)
 8:         else if $i = 0$ then
 9:             $former$ = None
10:             $latter$ = synsets($w_{i+1}$, $ld$)
11:         else
12:             $former$ = synsets($w_{i-1}$, $ld$)
13:             $latter$ = None
14:         for $s_c$,$s_f$,$s_l$ in (current, former, latter) do    ▷ Where $0 \leq c \leq Q, 0 \leq f \leq R$ and $0 \leq s \leq S$
15:             $a\_current \longleftarrow$ synset-vec($s_c$, $tsm$)
16:             $a\_former \longleftarrow$ synset-vec($s_f$, $tsm$)
17:             $a\_latter \longleftarrow$ synset-vec($s_l$, $tsm$)
18:             $t$ = best_score($a\_current$, $a\_former$, $a\_latter$)    ▷ Where the cosine similarity is performed between ($a\_current$,$a\_former$) and ($a\_current$,$a\_latter$), returning the $synset$ with the highest score for $a\_current$
19:             list_of_tokens $\longleftarrow t$
20:     return $list\_of\_tokens(synsets)$
```

tor values for all synsets we calculate the similarity of them for the pairs ($a\_current$,a_former) and ($a\_current$,a_later) and return the synset for $a\_current$ that produces the best result (lines 18 and 19).

The hope is that, over many passes, the results will converge to some value. We can stop the process after a finite number of passes, when we are satisfied that the results do not change much, or when the cost incurred for running another pass of the algorithm is too high to justify another disambiguation and annotation round.

3.3 (13,3) Most Suitable Sense Annotation - Dijkstra (MSSA-D)

We also developed another variation for the MSSA algorithm, in which we model the documents in the corpus as graphs $Doc_k(N, E)$, where Doc_k is the set of k documents; N is the set of nodes, represented by word-senses (synsets) and E is the set of edges associating two nodes. Inspired by Dijkstra's algorithm (1959), we use a modified version of it to minimize the over all cost of moving from one node (synset) to another for all the words in the document. The weights on the edges are represented by the *cosine distance* (1 - *cosine similarity*) between the gloss-average-vector of two sequential word-senses, which help to select one word-sense from the ones available. All the steps in MSSA-D design are the same as the ones presented in Section 3.1 for MSSA (Algorithm 1), with the exception there is no sliding context window for the disambiguation part. Different from MSSA, in the MSSA-D we analyze the disambiguation problem globally, looking for the shortest path from one word-sense to the next. Figure 3.3 illustrates a toy example of five words in which the highlighted path indicates the lowest cost considering their word-senses $\omega_{n,m}$, where n is the word position associated and m its respective sense. In the end, the objective of this algorithm is the same as the ones presented in Sectionsr̃efssec:mssa and 3.2, transform a training corpus composed by words into a corpus of synsets to be trained by word2vec.

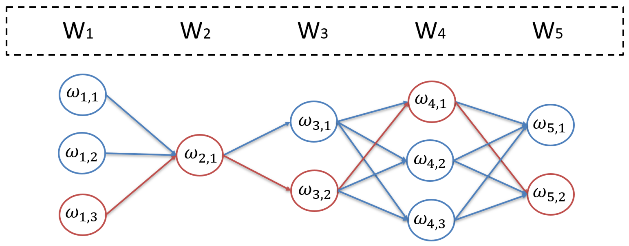

Figure 2: MSSA-D illustration of the shortest path from w_1 to w_5 through their respectives word-senses.

As in MSSA, it is also possible to apply MSSA-NR recurrent methodology into MSSA-D and use the produced synset embeddings instead of the Google News vector one. In Section 5, we describe the different setups used in our experiments to explore our techniques.

3.4 Synset to Embeddings (Synset2Vec)

Once all words are properly processed into synsets, we feed a word2vec implementation using the CBOW as training algorithm. This choice is justified due to of its popularity among the compared systems, reported superiority in performance and capturing the context of words in large datasets (Yu and Dredze, 2014; Yao et al., 2017; Bojanowski et al., 2017).

To keep our vectors interpretable - as pointed out by Pachenko (2016) - across different platforms, we represent each word token as key in the following format: $word\#synset\_offset\#pos$; where $word$ is the word itself, normalized and in lower case; $synset\_offset$ is an 8 digit, zero-filled decimal integer that corresponds to a unique word-sense and pos is a part-of-speech tag (e.g. n for nouns, v for verbs, a for adjective, s for adjective satellite and r for adverb)[7]. Since we have independent tasks for annotation and word embeddings training, if a more robust technique is proposed in the future, we can easily change to it.

4 (12) Multi-Sense Embeddings Measures

In a standard n-dimensional vector representation (e.g. word2vec), there will be just one embedding for each token to calculate any similarity measure between them. In a multi-vector representation, each word is associated with a set of senses, each with a vector representation. Hence, the similarity of these word-senses need to be calculated in a different way. Both representations make use of several benchmarks to evaluate how good they are in WS tasks. These benchmarks can be grouped into two categories, with and without context information. In the first, a similarity score is given for two words in isolation, without any extra information about them. In the second, each word is presented with a sentence to help contextualize its semantic value.

Considering the multi-vector representation, two metrics were initially proposed by Reisinger and Mooney (2010b): *AvgSim* and *MaxSim*. In AvgSim, word similarity is calculated by considering the average similarity of all word-sense embeddings for the pair, as shown in Equation 1.

$$AvgSim(u,w) = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} d(e(u,i),e(w,j)) \quad (1)$$

where u and w are the words to be compared; N and M are the total number of available senses for u and v respectively; $d(e(u,i),e(w,j))$ is the similarity measure between the word-sense embeddings sets denoted by $e(u,i)$ and $e(w,j)$, for the words u and w, and each of their senses i and j. In this paper, all similarity measures are calculated using the *cosine similarity* between any two vectors. In MaxSim, the similarity is the maximum value among all pairs of word-sense embeddings, as illustrated in Equation 2

$$MaxSim(u,w) = \max_{1\leq i\leq N, 1\leq j\leq M} d(e(u,i),e(w,j)) \quad (2)$$

(28)Reisinger and Mooney (2010b) also proposed *AvgSimC* and *MaxSimC*, these take into account the similarity of two words when their context is available. In this scenario, the context is represented by a sentence where each word is used, this helps to better illustrate the meaning of them. Both, AvgSimC and MaxSimC are described in Equations 3 and 4 respectively.

$$AvgSimC(u,w) = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} P(u,c_u,i) \\ P(w,c_w,j) \times d(e(u,i),e(w,j)) \quad (3)$$

$$MaxSimC(u,w) = d(e_k(u,i),e_k(w,j)) \quad (4)$$

(28)where $P(u,c_u,i) = d(e(u,i),c_u)$, is defined as the similarity of a specific word-sense $e(u,i)$ with its context c_u, which is obtained by averaging the vectors of all words in the sentence that accompanies each word u and w. Different from single word vector representations, our model produces vectors for each word-sense, so we also consider all available vectors for the words in the context sentence as well. $e_k(u,i) = \arg\max d(e(u,i),c_u)$ is the maximum similarity obtained among all word-senses $e(u,i)$, with respect to its context c_u. All these terms are defined analogously for w and j as well.

Huang et al., (2012) argued that word representations should be discriminated considering their surrounding words (*local context*) and their role in the entire document (*global context*). Their training model produces two vector types, one representing each word-sense and another for the word in the entire document, evaluated through *LocalSim* and *GlobalSim* respectively (Neelakantan et al., 2014). Different from Huang et al., (2012) and Neelakantan et al., (2014) our approach does not produce

[7]`https://wordnet.princeton.edu/documentation/wndb5wn`

global vectors during the training step, only specific ones. Therefore, to obtain a global representation of a word we average all word-sense vectors of u and w available to calculate their similarity, as Equation 5 shows.

$$GlobalSim(u, w) = d(\check{\mu}(u, i), \check{\mu}(w, j)) \tag{5}$$

where $\check{\mu}(u, i)$ and $\check{\mu}(w, j)$ represent the average of all word-sense vectors for u and w. As LocalSim, we can use the original MaxSimC instead, since they work under the same assumptions (Reisinger and Mooney, 2010b; Neelakantan et al., 2014).

5 Word Similarity Experiments

We designed a series of experiments for the WS task to evaluate how our algorithms compare against other approaches in the literature. In the next section we will provide details on the training corpus, benchmarks used and assessed systems.

5.1 Training Corpus

We applied our MSSA algorithms in two datasets to transform their words into synsets, using WN 3.1 (Fellbaum, 1998) as our lexical database. These datasets are Wikipedia Dumps composed by wiki articles from April 2010 (WD10) (Shaoul and Westbury, 2010) and January 2018 (WD18). While WD10 is commonly used as a training corpus in many WS tasks (Huang et al., 2012; Neelakantan et al., 2014; Iacobacci et al., 2015; Li and Jurafsky, 2015; Chen et al., 2015; Liu et al., 2015), WD18 is introduced by us as a variation in our experiments. Table 1 shows the details for both training sets in our experiments.

POS	**Words** (10^6)		**Synsets**	
	WD10	**WD18**	**WD10**	**WD18**
Nouns	299.41	463.31	55731	56546
Verbs	130.14	161.96	11975	12237
Adverbs	27.25	31.17	3091	3056
Adjectives	75.77	104.03	15512	15798
Total	532.57	760.47	86309	87637

Table 1: Dataset token details. WD10 - Wikipedia Dump 2010 (April); WD18 - Wikipedia Dump 2018 (January).

5.2 Hyperparameters, Setup and Details

Once all words in the training corpus are processed into synsets, we use a word2vec's implementation to produce our synset embeddings. The hyperparameters are set as follows: CBOW for the training algorithm, window size of 15, word minimum count of 10, hierarchical softmax and vector sizes of 300 and 1000 dimensions. If not specified, all the other hyperparameters were used with their default value[8]. Our system was implemented using Python 3.6.5, with NLTK (*Natural Language Toolkit*) 3.2.5, using the *gensim* 3.4.0[9] (Řehůřek and Sojka, 2010) library.

(32,33,22)In our experiments, we evaluate our approach with several systems described in Sections 5.4 and 5.5, using two different training datasets (WD10 and WD18) for the WS task as a whole. However, in a secondary analysis we also explored the properties of our models separately through different perspectives. For WD10, we discuss the effects of the number of iterations on MSSA-NR, with $-N$ ranging from 0 to 2. $N = 0$ characterizes the initial scenario in Figure 3, in which we use Google News vectors for the disambiguation step. For WD18, we investigate which of our representations of a word-sense performs better in the task, the one considering a local context (MSSA) or a global one (MSSA-D). The comparison of our recurrent model (MSSA-NR) against the MSSA-D algorithm is not explored in the proposed experiments, but we plan to include it in the nearest opportunity. To analyze how our synset embeddings are affected by the timestamp difference in the Wikipedia snapshot we do compare the results of MSSA for both training corpus. The number of dimensions used in our experiments is 300, where there is no specific label, and 1000 indicated with $-T$ next to the algorithm's name.

(26)The differences between metrics names, benchmarks and hyperparameters make it difficult to perform a direct comparison between systems. Recent publications also pointed out several problems (e.g. model overfitting, subjectivity) in using WS tasks to evaluate word embeddings approaches (Berg-Kirkpatrick et al., 2012; Faruqui et

[8]`https://radimrehurek.com/gensim/models/word2vec.html`

[9]`https://radimrehurek.com/gensim/`

al., 2016). We try to mitigate this situation by considering some aspects, such as: general perspective of proposed architecture, detailed description of the components used, training corpus specification and hyperparameters. In addition, we also categorize the reported results by other systems according to the correct metrics (AvgSim, AvgSimC, MaxSim, MaxSimC/LocalSim and GlobalSim) defined by seminal authors (Reisinger and Mooney, 2010b; Huang et al., 2012).

The results presented in Sections 5.4 and 5.5 are organized in three blocks for each benchmark (Table 2: 7), where they are separated by a break line and ordered as follows:

1. **Single-sense embeddings**: traditional word embeddings where all word-senses are collapsed into one vector representation per word. Approaches that concatenate a word vector with its sense are also included;

2. **Multi-sense embeddings**: each word-sense has a specific vector representation. Approaches that have a vector for both the word and its senses separately are also included;

3. **MSSA embeddings**: groups all our proposed models, only composed by multi-sense embeddings.

Results not reported by the authors in their systems are marked as "-" for the given metrics.

With the exception of MSSG (Neelakantan et al., 2014), (Chen et al., 2014) and CNN-MSSG (Chen et al., 2015), which aretrained using the SG model, all the compared systems either use CBOW or an independent approach of word embeddings.

All proposed algorithms (MSSA, MSS-D, MSSA-NR), corpora and models used in this paper are available in a public repository [10].

5.3 Benchmarks Details

The experiments were separated into two major categories, based on the datasets' characteristics: No Context Word Similarity (NCWS) and Context Word Similarity (CWS). The former (from 1 to 5), groups benchmarks used in WS tasks and provides similarity scores for word pairs in isolation. The latter (6), is composed by only one dataset that provides a collection of word pairs with their similarity scores. Each word is followed by sentences in which they are used to illustrate a context where they are applied. The benchmarks used are described as follows:

1. **RG65**: 65 noun pairs. Similarity scale from 0 to 4 (Rubenstein and Goodenough, 1965);

2. **MC28**: 28 pairs of nouns that were chosen to cover high, intermediate, and low levels of similarity in RG65. Similarity scale from 0 to 4 (Miller and Charles, 1991; Resnik, 1995);

3. **WordSim353**: 353 noun pairs divided into two sets of English word pairs. The first set contains 153 word pairs (similarity) and the second 200 (relatedness). As in compared systems, we use the concatenation of both sets. Similarity scale from 0 to 10 (Finkelstein et al., 2002);

4. **MEN**: 3,000 word pairs, randomly selected from words that occur at least 700 times in the ukWaC and Wacky corpora combined, and at least 50 times in the *ESP Game*. Similarity scale from 0 to 50 (Bruni et al., 2012);

5. **SimLex999**: 666 noun-noun pairs, 222 verb-verb pairs and 111 adjective-adjective pairs. Similarity scale from 0 to 10 (Hill et al., 2015);

6. **SCWS - Stanford Context Word Similarity**: 2,003 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. Similarity scale from 0 to 10 (Huang et al., 2012).

We tried to keep our basic configuration as close as possible to recent previous publications, so we considered the cosine similarity as our similarity measure and report the Spearman correlation value (ρ) in our experiments. To guarantee a common scenario between all benchmarks we normalized their similarity scale to an interval of [-1, 1]. Very few publications reported results for both Spearman and Pearson correlation values, but we considered the

[10] `to-be-provided-upon-acceptance`

first only, so more systems could be included in our paper. The results reported in our experiments, for all model variations, have high significant Spearman order correlation, with $p - value$ under 0.001.

Despite its similarity with our approach, one limitation in our analysis is regards SensEmbed (Iacobacci et al., 2015). SensEmbed uses the *Tanimoto* distance for their vector comparison, which has different properties from cosine similarity/distance used by the other compared systems. Therefore, even though SensEmbed presents good results, we did not include them in our analysis.

5.4 No Context Word Similarity

In this section, we evaluate our model against popular approaches available for 5 benchmarks: RG65, MEN, WordSim353, SimLex999 and MC28. We compare our results with: Pruned-TF-IDF (Reisinger and Mooney, 2010a), SW2V (using BN and WN with UMBC and Wikipedia Dump from 2014) (Mancini et al., 2017), DeConf (Pilehvar and Collier, 2016), Retro (using Glove with 6 billion words and WN with all synsets) (Faruqui et al., 2015), Glove (using 6 and 42 billion words) (Pennington et al., 2014), Word2vec (Mikolov et al., 2013b), (Chen et al., 2014), Word2vec (using UMBC and WD14) (Mancini et al., 2017), (Huang et al., 2012) and (NP)MSSG (for 50 and 300 dimensions) (Neelakantan et al., 2014).

Table 2 shows the results of MSSA against several models for the RG65 benchmark, in which DeConf-Sense (Pilehvar and Collier, 2016) presents the highest result considering the MaxSim measure. However, our model MSSA-2R-F hold superior scores for AvgSim and GlobalSim (GloSim). Different from DeConf, our model is capable to build on itself in every iteration by using the synset embeddings produced in the previous cycle, improving its performance. This can be verified by the constant increase in the Spearman correlation values reported in our approaches (MSSA), using the WD10 dataset for all three measures. The same two models are also in the top results for the MEN benchmark in Table 3. For this dataset, it seems that our model converges quicker since there are no improvements from MSSA-1R to MSSA-2R. We were only able to see some progress when we moved from 300 to 400 dimensions (MSSA-2R-F). The increase of di-

Models	**Avg Sim**	**Max Sim**	**Glo Sim**
GloVe-42B	-	-	0.829
GloVe-6B	-	-	0.778
Retro-G6B	-	-	0.767
Retro-G6B-WN	-	-	0.842
Word2Vec	-	-	0.754
DeConf-Sense	-	0.896	-
DeConf-Word	-	0.761	-
SensEmbed	0.871	0.894	-
SW2V-Shallow	-	0.740	-
SW2V-Babelfy	-	0.700	-
MSSA(WD10)	0.779	0.857	0.830
MSSA-1R(WD10)	0.795	0.872	0.825
MSSA-2R(WD10)	0.814	0.869	0.858
MSSA-T(WD10)	0.783	0.878	0.845
MSSA-1R-T(WD10)	0.825	0.871	0.856
MSSA-2R-T(WD10)	0.822	0.878	0.859
MSSA(WD18)	0.828	0.794	0.821
MSSA-D(WD18)	0.801	0.826	0.817
MSSA-T(WD18)	0.776	0.847	0.816
MSSA-D-T(WD18)	0.795	0.839	0.835

Table 2: Spearman correlation score (ρ) on RG65 benchmark. Highest results reported in **bold** face.

Models	**Avg Sim**	**Max Sim**	**Glo Sim**
Retro-G6B	-	-	0.737
Retro-G6B-WN-All	-	-	0.759
Word2vec(UMBC)	-	0.750	-
Word2vec(WD14)	-	0.720	-
Chen et al.,(2014)	-	0.62	-
DeConf-Sense	-	0.786	-
DeConf-Word	-	0.732	-
SensEmbed	0.805	0.779	-
SW2V-BN-UMBC	-	0.75	-
SW2V-WN-UMBC	-	0.76	-
SW2V-BN-WD14	-	0.73	-
SW2V-WN-WD14	-	0.72	-
MSSA(WD10)	0.751	0.745	0.760
MSSA-1R(WD10)	0.781	0.751	0.790
MSSA-2R(WD10)	0.777	0.737	0.788
MSSA-T(WD10)	0.778	0.753	0.785
MSSA-1R-T(WD10)	0.783	0.747	0.791
MSSA-2R-T(WD10)	0.785	0.744	0.795
MSSA(WD18)	0.745	0.769	0.775
MSSA-D(WD18)	0.768	0.716	0.765
MSSA-T(WD18)	0.769	0.749	0.776
MSSA-D-T(WD18)	0.772	0.717	0.767

Table 3: Spearman correlation score (ρ) on MEN benchmark. Chen et al., (2014) and Word2Vec results were reported by Mancini et al., (2017). Highest results reported in **bold** face.

mensionality seems to have a positive effect in most word embeddings models, including ours.

In Table 4, all results perform worse than GloVe (Pennington et al., 2014) for the GloSim measure for the GloSim metric. However, to reach this score they need to process 42 billion tokens, while when considering just 6 billion its performance decreases 13.30%. We, in the other hand, with a little less than 540 (WD10) million tokens can obtain superior results with MSSA-2R-F. Even though Pruned-TF-IDF (Reisinger and Mooney, 2010a) shows a competitive Spearman correlation value for GloSim, their model does not use low dimensionality vectors. In addition, its model relies on several parameter adjustments, such as: pruning cutoff, feature weighting, number of prototypes and feature representation. Under these circumstances, several of our models are ranked as top scores, with MSSA-2R-F and MSSA(WD18) among the highest results for AvgSim and MaxSim respectively.

Models	**Avg Sim**	**Max Sim**	**Glo Sim**
GloVe-42B	-	-	0.759
GloVe-6B	-	-	0.658
Retro-G6B	-	-	0.605
Retro-G6B-WN-All	-	-	0.612
Huang et al., (2012)	0.642		0.228
MSSG-50d	0.642	-	0.606
MSSG-300d	0.709	-	0.692
NP-MSSG-50d	0.624	-	0.615
NP-MSSG-300d	0.686	-	0.691
Pruned-TF-IDF	-	-	0.734
SensEmbed	0.779	0.714	-
SW2V-Shallow	-	0.710	-
SW2V-Babelfy	-	0.630	-
MSSA(WD10)	0.725	0.702	0.727
MSSA-1R(WD10)	0.711	0.661	0.712
MSSA-2R(WD10)	0.730	0.662	0.737
MSSA-T(WD10)	0.712	0.669	0.721
MSSA-1R-T(WD10)	0.708	0.666	0.716
MSSA-2R-T(WD10)	0.729	0.667	0.737
MSSA(WD18)	0.663	0.714	0.712
MSSA-D(WD18)	0.708	0.626	0.702
MSSA-T(WD18)	0.694	0.637	0.692
MSSA-D-T(WD18)	0.702	0.623	0.693

Table 4: Spearman correlation score (ρ) on WordSim353 benchmark. Huang et al., (2012) results were reported by Neelakantan et al., (2014). Highest results reported in **bold** face.

The last two NCWS benchmarks, SimLex999 and MC28, are particularly challenging for distinct reasons. For SimLex999 in Table 5, our models perform poorly regardless of their configuration, while DeConf presents the best results. The average Spearman correlation values for this dataset seems to be low in all publications, rarely surpassing $\rho = 0.50$. Even in our not reported models, we could not reach satisfactory results. The same behavior was observed when we tried to apply our model in only-verbs benchmarks, such as: YP130 (Yang and Powers, 2006) and SimVerb3500 (Gerz et al., 2016). For the former, our Spearman scores were on average $\rho = 0.563$, while for the latter $\rho = 0.243$ (MaxSim). Our suspicion is that our algorithm is not robust enough to deal with datasets of this nature. For MC28, reported in Table 6, the lack of recent publications makes it hard to draw any conclusions. If we consider ACL State-of-the-art Wiki[11] we would have obtained the third best result after the human upper bound.

Models	**Avg Sim**	**Max Sim**	**Glo Sim**
Word2vec(UMBC)	-	0.390	-
Word2vec(WD14)	-	0.380	-
Chen et al.,(2014)	-	0.430	-
DeConf-Sense	-	0.517	-
DeConf-Word	-	0.443	-
SW2V-BN-UMBC	-	0.470	-
SW2V-WN-UMBC	-	0.450	-
SW2V-BN-WD14	-	0.430	-
SW2V-WN-WD14	-	0.430	-
MSSA(WD10)	0.427	0.368	0.396
MSSA-1R(WD10)	0.438	0.369	0.405
MSSA-2R(WD10)	0.440	0.369	0.408
MSSA-T(WD10)	0.456	0.393	0.432
MSSA-1R-T(WD10)	0.468	0.394	0.441
MSSA-2R-T(WD10)	0.469	0.385	0.439
MSSA(WD18)	0.375	0.438	0.404
MSSA-D(WD18)	0.401	0.351	0.374
MSSA-T(WD18)	0.460	0.389	0.430
MSSA-D-T(WD18)	0.425	0.372	0.391

Table 5: Spearman correlation score (ρ) on SimLex999 benchmark. Chen et al., (2014) and Word2Vec results were reported by Mancini et al., (2017). Highest results reported in **bold** face.

In all benchmarks, with the exception of SimLex999, our proposed models (MSSA) sustain competitive results. Considering the MaxSim and GloSim, we report either the first or second best Spearman correlation values in the experiments. For the AvgSim, our approach exhibits the highest score in all datasets, including SimLex999.

5.5 Context Word Similarity

In this section the results reported by AutoExtend(Synsets) (Rothe and Schütze, 2015) and CNN-VMSSG (Chen et al., 2015) were also incorporated. For the SCWS benchmark, we did not report the re-

[11] `https://aclweb.org/aclwiki/Similarity_(State_of_the_art)`

Models	Avg Sim	Max Sim	Glo Sim
GloVe-42B	-	-	0.836
GloVe-6B	-	-	0.727
MSSA(WD10)	0.833	0.862	0.842
MSSA-1R(WD10)	0.825	0.883	0.843
MSSA-2R(WD10)	0.829	0.849	0.847
MSSA-T(WD10)	0.845	0.888	0.875
MSSA-1R-T(WD10)	0.841	0.883	0.862
MSSA-2R-T(WD10)	0.801	0.866	0.8363
MSSA(WD18)	0.775	0.799	0.792
MSSA-D(WD18)	0.835	0.807	0.829
MSSA-T(WD18)	0.796	0.834	0.818
MSSA-D-T(WD18)	0.801	0.833	0.821

Table 6: Spearman correlation score on MC28 benchmark. Highest results reported in **bold** face.

sults for the MaxSim measure, since almost all publications do not report them. The SCWS dataset provides pair of words given a specific context, alleviating dubious interpretations (Huang et al., 2012).

Table 7 shows DeConf (Pilehvar and Collier, 2016), MSSG (Neelakantan et al., 2014) and our model (MSSA) with the highest Spearman correlation values, in descending order for AvgSim. However, when considering MaxSimC and GloSim, MSSA and MSSA-2R hold state-of-the-art results. CNN-VMSSG presents the second highest results, but its model relies on two training steps instead of one. For AvgSimC results, it seems that our weighting scheme was not able to capture the nuances of the context for each word in a proper manner. It would be interesting to apply the top ranked algorithms to our model and compare their performance. Unfortunately, DeConf is designed to use a pre-trained single n-dimensional vector representation to produce their multi-sense embeddings, thus our approach would not be easily applicable. NP-MSSG (Neelakantan et al., 2014) and SW2V (Mancini et al., 2017) on the other hand, offer the necessary flexibility to use our annotated corpus to produce new embeddings.

Both MSSA and MSSA-D are trying to optimize a cost function, but in different ways. The former works in a local manner, minimizing its cost for the overlapping context window, one at a time. The latter finds the least costly path from the first to the last word-sense for the entire document. Initially, we thought that MSSA-D would produce the best result on average, since it considers the whole document as its global context. However, if we analyze

Models	Avg Sim	Avg SimC	Max SimC	Glo Sim
GloVe-42B	-	-	-	0.596
GloVe-6B	-	-	-	0.539
Autoextend	0.626	0.637	-	-
Chen et al., (2014)	0.662	0.689	-	0.642
CNN-VMSSG	0.657	0.664	0.611	0.663
DeConf-Sense	0.708	0.715	-	
Huang et al., (2012)	0.628	0.657	0.261	0.586
MSSG-50d	0.642	0.669	0.492	0.621
MSSG-300d	0.672	0.693	0.573	0.653
NP-MSSG-50d	0.640	0.661	0.503	0.623
NP-MSSG-300d	0.673	0.691	0.598	0.655
Pruned-TF-IDF	0.604	0.605	-	0.625
SensEmbed	-	0.624	0.589	-
MSSA(WD10)	0.667	0.581	0.637	0.667
MSSA-1R(WD10)	0.660	0.581	0.639	0.659
MSSA-2R(WD10)	0.665	0.585	0.646	0.665
MSSA-T(WD10)	0.659	0.590	0.617	0.664
MSSA-1R-T(WD10)	0.655	0.594	0.623	0.658
MSSA-2R-T(WD10)	0.661	0.604	0.617	0.634
MSSA(WD18)	0.593	0.569	0.639	0.651
MSSA-D(WD18)	0.640	0.557	0.613	0.640
MSSA-T(WD18)	0.649	0.588	0.617	0.654
MSSA-D-T(WD18)	0.638	0.570	0.597	0.639

Table 7: Spearman correlation score (ρ) on SCWS benchmark. Huang et al., (2012) results were reported by Neelakantan et al., (2014). Highest results reported in **bold** face.

the results from WD18 only, this is not consistent among all benchmarks. MSSA proved to capture the semantic relations more accurately in several cases (RG65, MEN, WordSim353 and SimLex) in the WS task, especially for the MaxSim measure. Apparently, the features of a local context are more discriminative than universal common characteristics.

6 Final Considerations

In this paper, we proposed a system called MSSA that automatically disambiguates and annotates any text corpus considering a sliding context window for each word. We have demonstrated that single vector representation limitations can be mitigated by applying MSSA into a traditional word2vec implementation, producing more robust multi-sense embeddings with minimum hyperparameter tuning. Additionally, we performed an extensive comparison with many recent publications in the WS task and categorized their results according to standard metrics (Section 4). The representation used in our model can be easily interpreted and extended to other NLP task beyond WS. Finally, we showed that the combination between the proposed MSSA algorithm and word2vec is able to sustain solid results in 6 different benchmarks: RG65, MEN, WordSim353, SimLex999, MC28 and SCWS. Currently, our model considers a sliding context window of +/- 1 tokens, unigrams and non-stemmed words, but we intend to pursue some extensions, such as: keep common n-grams, flexible context sliding window of size k and different weighting schemes for the context analysis. We also would like to integrate MSSA with the compared systems that obtained good results throughout our experiments.

References

Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in nlp. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 995–1005, Stroudsburg, PA, USA. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *In Proceedings of 15th International Conference on Artificial Intelligence and Statistics*.

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: Long Papers - Volume 1*, ACL '12, pages 136–145, Stroudsburg, PA, USA. Association for Computational Linguistics.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. NASARI: a novel approach to a semantically-aware representation of items. In *HLT-NAACL*, pages 567–577. The Association for Computational Linguistics.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artif. Intell.*, 240:36–64.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*, pages 1025–1035. ACL.

Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2015. Improving distributed representation of word sense via wordnet gloss composition and context clustering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 15–20. The Association for Computational Linguistics.

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 P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Paramveer S. Dhillon, Dean Foster, and Lyle Ungar. 2011. Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24.

E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *HLT-NAACL*, pages 1606–1615. The Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *CoRR*, abs/1605.02276.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, January.

J. R. Firth. 1957. A synopsis of linguistic theory 1930-55. 1952-59:1–32.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In *EMNLP*.

William I Grosky and Terry L Ruas. 2017. The Continuing Reinvention of Content-Based Retrieval: Multimedia Is Not Dead. *IEEE MultiMedia*, 24(1):6–11, jan.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *ACL (1)*, pages 95–105. The Association for Computer Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *ACL (1)*. The Association for Computer Linguistics.

Richard Johansson and Luis Nieto Piña. 2015. Embedding a semantic network in a word space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Denver, United States, May 31 June 5, 2015*, pages 1428–1433.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732. Association for Computational Linguistics.

H. Liu and P. Singh. 2004. Conceptnet &mdash; a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, October.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2418–2424. AAAI Press.

Massimiliano Mancini, José Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2017. Embedding words and senses together via joint knowledge-enhanced training. In *CoNLL*, pages 100–111. Association for Computational Linguistics.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language & Cognitive Processes*, 6(1):1–28.

George A. Miller. 1995. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69, February.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*, pages 1059–1069. ACL.

Alexander Panchenko. 2016. Best of both worlds: Making word sense embeddings interpretable. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Mohammad Taher Pilehvar and Nigel Collier. 2016. Deconflated semantic representations. In *EMNLP*, pages 1680–1690. The Association for Computational Linguistics.

Mohammad Taher Pilehvar, José Camacho-Collados, Roberto Navigli, and Nigel Collier. 2017. Towards a seamless integration of word senses into downstream NLP applications. *CoRR*, abs/1710.06632.

Luis Nieto Piña and Richard Johansson. 2015. A simple and efficient method to generate word sense representations. In *Proceedings of International Conference in Recent Advances in Natural Language Processing*, pages 465–472.

Hilary Putnam. 1970. Is semantics possible? *Metaphilosophy*, 1(3):187–201.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Joseph Reisinger and Raymond Mooney. 2010a. A mixture model with sharing for lexical semantics. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1173–1182, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joseph Reisinger and Raymond J. Mooney. 2010b. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 109–117, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *ACL (1)*, pages 1793–1803. The Association for Computer Linguistics.

Sascha Rothe and Hinrich Schütze. 2017. Autoextend: Combining word embeddings with semantic resources. *Computational Linguistics*, 43(3):593–617.

Terry Ruas and William Grosky. 2017. Keyword Extraction Through Contextual Semantic Analysis of Documents. In *Proceedings of the 9th International Conference on Management of Emergent Digital EcoSystems*, pages 150–156, Bangkok. ACM Press.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October.

Holger Schwenk, Daniel Dchelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 723–730, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cyrus Shaoul and Chris Westbury. 2010. The westbury lab wikipedia corpus.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 151–160.

Andrew Trask, Phil Michalak, and John Liu. 2015. sense2vec - A fast and accurate method for word sense disambiguation in neural word embeddings. *CoRR*, abs/1511.06388.

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*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *CoRR*, abs/1003.1141.

Dongqiang Yang and David M. W. Powers. 2006. Verb similarity on the taxonomy of wordnet. In *In the 3rd International WordNet Conference (GWC-06), Jeju Island, Korea*.

Yao Yao, Xia Li, Xiaoping Liu, Penghua Liu, Zhaotang Liang, Jinbao Zhang, and Ke Mai. 2017. Sensing spatial distribution of urban land use by integrating points-of-interest and google word2vec model. *Int. J. Geogr. Inf. Sci.*, 31(4):825–848, April.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *ACL (2)*, pages 545–550.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 78–83, Stroudsburg, PA, USA. Association for Computational Linguistics.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.