Retrofitting Word Vectors into Sense Level Representations

Anonymous Author(s)

ABSTRACT

Research in Natural Language Processing proposes many techniques for generating or retrofitting distributed representations by utilizing external semantic knowledge. This paper reports on such an approach for retrofitting pre-trained word representations into sense level representations. We use semantic relations as positive and negative examples to refine the results of a pre-trained model instead of integrating them into the objective functions used during training. We experimentally evaluate our approach on two word similarity tasks by retrofitting six datasets generated from three widely used techniques for word representation using two different strategies. Our approach significantly and consistently outperforms three state-of-the-art retrofitting approaches. It also minimizes the influence of vector dimensions and the size of the corpus at the expense of a few iterative refinements of the vectors.

KEYWORDS

natural language processing, post-processing model, retrofitting, word representations, knowledge-based sense representations, negative sampling

ACM Reference format:
Anonymous Author(s). 2019. Retrofitting Word Vectors into Sense Level Representations. In *Proceedings of Woodstock '18: ACM Symposium on Neural Gaze Detection, Woodstock, NY, June 03–05, 2018 (Woodstock '18),* 10 pages.
DOI: 10.1145/1122445.1122456

1 INTRODUCTION

Distributed word representations based on word vectors learned from distributional information about words in large corpora have become a central technique in Natural Language Processing (NLP) because of their excellent performance in many tasks, examples including sentiment analysis [33], question answering [60], or machine translation [7]. On basis of the distributional hypothesis [22], methods convert words into vectors by linguistic contexts as "predictive" models [4, 20, 34, 36] or by co-occurring words as "count-based" models [39]. Both of them depend on "co-occurrence" information on words in a large unlabeled corpus. In general, we can observe that the larger data they use, the better such methods tend to perform on NLP tasks.

These approaches for constructing vector spaces predominantly focus on contextual relationships or word morphology. That means they disregard the constraints obtainable from lexicons which provide semantic information by identifying synonym, antonym, hypernymy, hyponymy, and paraphrase relations. This impedes their performance on word similarity tasks and applications where word similarity plays a significant role such as, e.g., text simplification.

Existing approaches for exploiting external semantic knowledge to improve word vectors can be grouped into two categories [52]: (1) *joint specialization* models integrate semantic constraints on word similarity by modifying the objective of the original word vector training in joint neural language models [35, 59] or by incorporating relation-specific constraints like the co-occurrence matrix [11] or word ordinal ranking [31] into models; (2) *post-processing* models retrofit or refine the pre-trained distributional word vectors in order to fit the semantic constraints [14, 27, 47, 52].

Compared with joint specialization models, post-processing models, obviously, are more flexible because they can be applied to all kinds of distributional spaces. Furthermore, post-processing approaches do not need to re-train models on the large corpora typically used, which is more convenient both for research purposes and in applications.

Recent work on post-processing approaches, mainly based on the graph-based learning technique [14, 27, 58], has had a great influence on the field of retrofitting word vectors. Yet, these studies specifically show the significant improvements on benchmarks of evaluation datasets such as MEN [8] or WordSim-353 [16] which conflate relatedness or association with similarity rather than on datasets that exclusively focus on word similarity.

In this paper, we propose a new approach that obtains the new word vectors by retrofitting pre-trained vectors by exploiting synonyms and antonyms obtained from *thesaurus.com*[1] for the 100,000 most frequently-used English words from Wiktionary.[2] For evaluation purposes, we use the same standard benchmarks as used by Vulic and Glavas [52], i.e., SimLex-999 [23] and SimVerb-3500 [18], and also Stanford's Contextual Word Similarities [24] used by Lee et al. [27].

We find that our model provides significant improvements in word similarity compared to state-of-the-art retrofitting models. Rather than neural networks with complicated hidden layers [52], this approach uses negative sampling [36] for simplifying this part. The contributions of this paper are twofold: (i) we generate the vectors of a word with different definitions using the original pre-trained word vector, based on the synonym and antonym sets in the different word senses, and (ii) we simplify the complicated hidden layers of the neural network to enhance the performance and decrease the running time.

Figure 1 depicts the structure of our approach. We input the synonyms of a target word with the j-th sense into our model which are the positive samples giving a positive influence on this target word. The antonyms of this target word are added to adjust the probability of this target word. Such adjustments can also affect

Peter
Is "retrofitt
vectors" no
as "post-pr
Maybe "di
word repre
as the field

Rui
I am not su
are totally
think post
includes m
not only re

[1] https://www.thesaurus.com/
[2] https://gist.github.com/h3xx/1976236

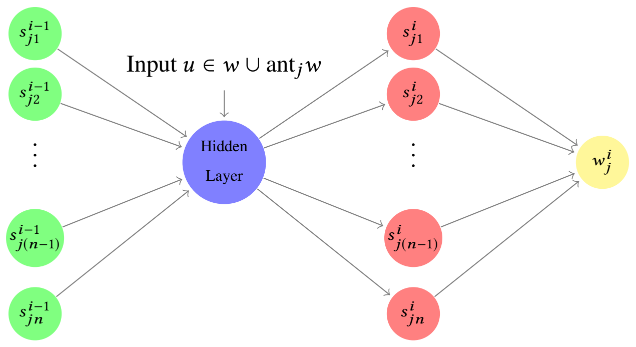

Figure 1: Graphical sketch of the proposed model. s_{jn}^i refers to the n-th synonym of the target word with the j-th sense in the i-th step and $ant(w_j)$ indicates the antonym set of the target word with the j-th sense.

these synonyms such that we could update the vectors of these synonyms first. Finally under the influence of the updated synonyms and the antonyms, the vector of the target word is retrofitted with the j-th sense.

This paper is structured as follows. An overview on recent related work is provided in Section 2 to display the background on word sense representation. In Section 3, we introduce the principle of our model, define the objective, and describe the steps of optimization as well as how to update word embeddings into sense embeddings. In Section 4 we describe our experimental setup with datasets, evaluation procedures, and the evaluated tasks. In Section 5 we discuss the results of different configurations on the benchmarks. Finally, we present conclusions and future challenges in Section 6.

2 RELATED WORK

The representation of words which provide continuous low-dimensional vector representations of words plays a pivotal role and has been widely studied in NLP domain. Vector Space Models (VSM) is the basis of many prominent methodologies for word representation learning. The earliest VSMs considered a vector space in document level which uses the vocabulary directly as the features [45]. Subsequently many kinds of weight computation metrics of individual dimensions such as word frequencies or normalized frequencies [44] have been proposed. This research has succeeded in various NLP tasks.

However, one crucial problem with the huge corpus is the high dimensionality of the produced vectors. A common solution is dimensionality reduction making use of the Singular Value Decomposition (SVD). Learning low-dimensional word representations directly from text corpora is another strategy that has been achieved by leveraging neural networks. These models are commonly known as word embeddings. Some prominent word embedding architectures have been constructed depending on contextual relationships or word morphology. Beyond that, more complex approaches have been proposed attempting to cure some deficiencies (e.g., conflation of meaning) by exploiting subword units [54], probability distributions [1], specialized similarity measures [48], knowledge resources [9], etc.

The process of producing word embeddings is not able to capture different meanings of the same word. And for downstream tasks, the meaning conflation can have a negative influence on accurate semantic modeling, e.g., one word may have several different senses that would be similar to several unrelated words in the different semantic space, like "*mouse-screen*" and "*mouse-cow*". There is apparently no relation between "*screen*" and "*cow*", but they can be connected by two different senses of "*mouse*", i.e., computer device and animal.

Partitioning the meanings of words into multiple senses is not an easy task. Computational approaches relying on text corpora or semantic knowledge (such as a dictionary or thesaurus), generally can be categorized into unsupervised techniques and knowledge-based techniques.

2.1 Unsupervised Techniques

Unlabeled monolingual corpora can be exploited for word sense disambiguation by clustering-based approaches or joint models. Clustering the context in which an ambiguous word occurs can discern senses automatically. Context-group discrimination [46] is an approach to compute the centroid vector of the context of an ambiguous word, and then cluster these context centroid vectors into a predetermined number of clusters. Context clusters for an ambiguous word are interpreted as representations for different senses of this word. Models applying this strategy are also called two-stage models [13, 50].

Joint models [29, 41] are proposed as various extensions of traditional word embedding models. The primary difference in contrast to clustering-based approaches is that joint models merge the clustering and the sense representation step. In this way, the joint model can dynamically select the potential sense for an ambiguous word during training. Topical Word Embeddings (TWE) [32] are proposed for inducing the sense representations of a word based only on its local context, which reduces computational complexity. Joint models have serious limitations, though, which require the disambiguation of the context of a word as well as predetermining a fixed number of senses per word.

Another recent branch of unsupervised techniques is to generate contextualized word embeddings. Here, context-sensitive latent variables for each word are inferred from a fuzzy word clustering and then integrated to the sequence tagger as additional features [30].

2.2 Knowledge-based Techniques

Knowledge-based techniques about semantic representations fall into three categories: (1) improving word representations, (2) using sense representations, and (3) using concept and entity representations. Studies related to all three categories make use of similar knowledge resources. In particular, WordNet [2] as an example of expert-made resources and Wikipedia as an example of collaboratively-constructed resources are widely applied [9]. Some similar collaborative works powered by Wikipedia like Freebase [5] and DBpedia [3] also provide large structured data in the form of the knowledge base. Further examples are BabelNet [37], a combination of expert-made resources and collaboratively-constructed resources, and ParaPhrase DataBase (PPDB) [17] gathering over 150 million paraphrases and providing a graph structure. In the following, we shortly recap some of the relevant related work for each of the three categories.

Improving Word Representations. The earlier attempts to improve word embedding using lexical resources modified the objective functions of a neural network model for learning a word representation [25, 55, 59]. Typically, they integrate the external semantic constraints into the learning process directly, resulting in joint specialization models. Some recent approaches try to improve the pre-trained word vectors through post-processing [14, 28], which is a more versatile approach than the joint models. The popular term "retrofitting" is used when implanting external lexicon knowledge into random pre-trained word vectors. Given any pre-trained word vectors, generated by any tools or techniques, the main idea of graph-based retrofitting is to minimize the distance between synonyms and maximize the distance between antonyms. Building upon the retrofitting idea, explicit retrofitting constructs a neural network by modeling pairwise relations [19, 52], which also specialize vectors of words unseen in external lexical resources.

Using Sense Representations. The second category consists of sense vector representation techniques. These generate vectors by "de-conflating" a word (with conflated meanings) into several individuals with different senses. Methods based on linear models draw support by the synonym and antonym sets of the target word with different senses to retrofit the original word vectors [27, 40]. Chen et al. [12] exploit a convolutional neural network architecture for sense embedding. Neelakantan et al. [38] rely on the Skip-gram model for learning sense embeddings. The Lesk algorithm [51] has been adapted for learning word sense embeddings [56].

Using Concept and Entity Representations. The main idea in this branch is to construct a strong relation between related entities. Given a knowledge base as a set of triples $\{(e_1, e_2, r)\}$, where e_1 and e_2 are entities and r is the relation between them, the goal is to approach the entities by the relation r ($\vec{e_1} + \vec{r} \approx \vec{e_2}$) for all triples in the space. Typical approaches integrating concepts and entities from external knowledge bases rely exclusively on knowledge graphs to build an embedding space for entities and relations [6]. In addition, some hybrid models have been proposed to exploit text corpora and knowledge bases as well [10].

2.3 Summary

We find that in prior work, models using neural networks are more accurate while linear approaches for retrofitting are more efficient and straightforward. In contrast, the approach we propose is inspired by Word2Vec [34, 36] and takes advantage of neural network learning but reduces the effect of the hidden layer as much as possible. The result is an approach that is both simple and practical. In addition to accuracy, we also consider sense representations in this paper to better deal with meaning conflation. In particular, we retrofit the pre-trained word vectors by external lexicon knowledge into sense level representations.

3 NEGATIVE-SAMPLING RETROFITTING

An important aspect in Natural Language Processing is the Statistical Language Model which can be used to calculate the probability of a sentence over a vocabulary of size N:

$$\Pr(w_1 w_2 \dots w_N).$$

We assume $w_1^N = (w_1 w_2 \dots w_N)$. According to Bayes' theorem, this probability could be decomposed into conditional probabilities:

$$\Pr(w_1), \Pr(w_2|w_1), \dots, \Pr(w_N|w_1^{N-1})$$

which are the parameters of a Language Model. In order to find the optimal model parameters, we will do optimization on an objective function $\Pr(w|\,\mathrm{POS}(w))$ generated by a maximum likelihood estimate method, where $\mathrm{POS}(w)$ is the positive sample set of the target word w. That is, under the positive condition, the probability of w should be greater than under the negative condition. Thus, we explore a method that uses a set of linguistic constraints from an external lexical resource,

$$LC(w_i) = \{w_i, \mathrm{syn}(w_i), \mathrm{ant}(w_i)\},$$

each consisting of a word $w_i \in \mathcal{V}$ ($\mathcal{V} = \{w_i\}_{i=1}^N$ referring to the associated vocabulary) including all senses with the corresponding synonyms ($\mathrm{syn}(w_{ij})$ with j-th sense) as positive samples and its antonyms ($\mathrm{ant}(w_i)$) as negative samples to retrofit the vector of each target word. More specifically, the synonyms and antonyms of the corresponding senses helps us to refine the vectors into a sense level. We employ Negative Sampling [36], a simplified version of Noise Contrastive Estimation (NCE) [21] which aims at improving the quality of results and decreasing training time.

Our approach consists of two major components: (1) updating the synonyms of the target word by negative sampling using the antonyms of this word, and (2) generating the corresponding new sense embedding of this word by its updated synonyms in the j-th sense.

Peter
Do we imp
quantify o
Existential
sally?

Rui
the numbe
for differen
different, s
indicating

3.1 Objective Functions

Let $X = \{\boldsymbol{x}_i\}_{i=1}^N$, $\boldsymbol{x}_i \in \mathbb{R}^d$ be the pre-trained d-dimensional distributed vector space and let $Y = \{\boldsymbol{y}_i\}_{i=1}^N$, $\boldsymbol{y}_i = \{\boldsymbol{y}_{w_{ij}}\}_{j=1}^{|sense_{w_i}|}$, be the corresponding retrofitted sense vector space. Recall that $\mathrm{syn}(w)$ is the positive sample set and $\mathrm{ant}(w)$ is the negative sample set of target word w, respectively. Therefore, for any word u in the pre-trained word vector space, we first define Equation (1) to denote the label of word u. That means if the word u is the target word, the corresponding optimization process for the objective function will be activated.

$$L^w(u) = \begin{cases} 1, & u = w \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

According to the given positive sample set $\mathrm{syn}(w)$, the objective is to maximize the conditional probability of a word under a condition of its synonyms. Certainly, each synonym can affect this condition, either on its own or with other related synonyms. On the basis of the above, we propose two strategies for this goal. One would be to maximize a series of probability functions if we set each synonym as an individual (NS-sv):

$$\begin{aligned} \mathcal{L} &= \prod_{w \in \mathcal{V}} \prod_{\widetilde{w} \in \mathrm{syn}(w)} g(w, \boldsymbol{\theta}) \\ &= \prod_{w \in \mathcal{V}} \prod_{\widetilde{w} \in \mathrm{syn}(w)} \prod_{u \in \{w\} \cup \mathrm{ant}(w)} \Pr(u|\widetilde{w}) \end{aligned} \tag{2}$$

where $\boldsymbol{\theta}$ is the parameter matrix of the hidden layer, and g is the objective conditional probability function.

An alternative strategy is to maximize the conditional probability under the integration of the synonyms (NS-sv-Sum):

$$\begin{aligned}\mathcal{L} &= \prod_{w\in\mathcal{V}} g(w,\boldsymbol{\theta})\\ &= \prod_{w\in\mathcal{V}} \prod_{u\in\{w\}\cup\mathrm{ant}(w)} \Pr(u|\widetilde{w})\end{aligned} \tag{3}$$

where $\widetilde{w}$ is composed of all synonyms instead of each single word. In our approach, we integrate them by:

$$\widetilde{w} = \sum_{s\in\mathrm{syn}(w)} s \tag{4}$$

3.2 Optimization

No matter which strategy we use to produce the objective, the goal is to maximize the probability of the positive sample and, simultaneously, to minimize the probability of the negative sample. We use a sigmoid function as the activation function:

$$\sigma(x) = \frac{1}{1+\exp(x)} \tag{5}$$

such that the derivative of sigmoid function is:

$$\sigma'(x) = \sigma(x)\,[1-\sigma(x)] \tag{6}$$

Thence, $\Pr(u|\widetilde{w})$ could be denoted as:

$$\Pr(u|\widetilde{w}) = \begin{cases}\sigma\left(\boldsymbol{x}_{\widetilde{w}}^{\mathrm{T}}\boldsymbol{\theta}^u\right), & L^w(u)=1\\ 1-\sigma\left(\boldsymbol{x}_{\widetilde{w}}^{\mathrm{T}}\boldsymbol{\theta}^u\right), & L^w(u)=0\end{cases} \tag{7}$$

Putting the pieces together, $g(w,\boldsymbol{\theta})$ would become:

$$g(w,\boldsymbol{\theta}) = \prod_{u\in\{w\}\cup\mathrm{ant}(w)} \sigma\left(\boldsymbol{x}_{\widetilde{w}}^{\mathrm{T}}\boldsymbol{\theta}^u\right)^{L^w(u)} \cdot \left(1-\sigma\left(\boldsymbol{x}_{\widetilde{w}}^{\mathrm{T}}\boldsymbol{\theta}^u\right)\right)^{1-L^w(u)} \tag{8}$$

The purpose of training by negative sampling is to maximize the objective with respect to the model parameters by the commonly used log-likelihood function:

$$\begin{aligned}\log g(w,\boldsymbol{\theta}) = \sum_{u\in\{w\}\cup\mathrm{ant}(w)} \Big\{ & L^w(u)\cdot\log\left[\sigma\left(\boldsymbol{x}_{\widetilde{w}}^{\mathrm{T}}\boldsymbol{\theta}^u\right)\right] +\\ & [1-L^w(u)]\cdot\log\left[1-\sigma\left(\boldsymbol{x}_{\widetilde{w}}^{\mathrm{T}}\boldsymbol{\theta}^u\right)\right]\Big\}\end{aligned} \tag{9}$$

To explain the procedure in detail, we let F_w^{kj} represent the log-likelihood function of the target word w in the k-th step with the j-th sense:

$$\begin{aligned}F_w^{kj} = & L_{kj}^w(u)\cdot\log\left[\sigma\left(\boldsymbol{x}_{\widetilde{w}}^{\mathrm{T}}\boldsymbol{\theta}_{kj}^u\right)\right] +\\ & \left[1-L_{kj}^w(u)\right]\cdot\log\left[1-\sigma\left(\boldsymbol{x}_{\widetilde{w}}^{\mathrm{T}}\boldsymbol{\theta}_{kj}^u\right)\right]\end{aligned} \tag{10}$$

We choose gradient ascent for optimization. The parameters $\boldsymbol{\theta}_{kj}^u$ and $\boldsymbol{x}_{\widetilde{w}}$ are updated by the learning rate α as follows:

$$\begin{aligned}\boldsymbol{\theta}_{kj}^u &= \boldsymbol{\theta}_{kj}^u + \alpha\frac{\partial F_w^{kj}}{\partial\boldsymbol{\theta}_{kj}^u}\\ \boldsymbol{x}_{\widetilde{w}} &= \boldsymbol{x}_{\widetilde{w}} + \alpha\sum_{u\in\{w\}\cup\mathrm{ant}_j(w)}\frac{\partial F_w^{kj}}{\partial\boldsymbol{x}_{\widetilde{w}}}\end{aligned} \tag{11}$$

Algorithm 1 NS-sv

```
Input:
    a pre-trained word embedding $X$;
    the associated vocabulary $\mathcal{V}$;
    the linguistic constraints $LC$;
    the parameter matrix $\boldsymbol{\theta}$;
    the learning rate $\alpha$;
Output:
    a retrofitted sense embedding $Y$;
1:  for each $w \in \mathcal{V}$ do
2:    find $LC(w) = w, \mathrm{syn}(w), \mathrm{ant}(w)$;
3:    for each $w_j \in LC(w)$ do
4:      for each $\widetilde{w} \in \mathrm{syn}(w_j)$ do
5:        for $u \in \{w\} \cup \mathrm{ant}(w)$ do
6:          $z = g(w, \boldsymbol{\theta})$;
7:          $p = F_w^{kj}$;
8:          $g = \alpha(p - z)$;
9:          $\boldsymbol{v} := \boldsymbol{v} + g\boldsymbol{\theta}_{kj}^u$;
10:         $\boldsymbol{\theta}_{kj}^u := \boldsymbol{\theta}_{kj}^u + g\boldsymbol{x}_{\widetilde{w}}$;
11:       end for
12:       $\boldsymbol{x}_{\widetilde{w}} := \boldsymbol{x}_{\widetilde{w}} + \boldsymbol{v}$;
13:     end for
14:     $\boldsymbol{y}_{w_j} := \boldsymbol{x}_w + \sum_{\widetilde{w}\in\mathrm{syn}(w_j)} \boldsymbol{x}_{\widetilde{w}}$
15:   end for
16: end for
```

The gradients above are calculated as follows:

$$\begin{aligned}\frac{\partial F_w^{kj}}{\partial\boldsymbol{\theta}_{kj}^u} &= \left[F_w^{kj} - \sigma\left(\boldsymbol{x}_{\widetilde{w}}^{\mathrm{T}}\boldsymbol{\theta}_{kj}^u\right)\right]\boldsymbol{x}_{\widetilde{w}}\\ \frac{\partial F_w^{kj}}{\partial\boldsymbol{x}_{\widetilde{w}}} &= \left[F_w^{kj} - \sigma\left(\boldsymbol{x}_{\widetilde{w}}^{\mathrm{T}}\boldsymbol{\theta}_{kj}^u\right)\right]\boldsymbol{\theta}_{kj}^u\end{aligned} \tag{12}$$

Consequently, we obtain the updating function for the parameter matrix and the synonym embeddings.

3.3 Generating Sense Vectors

In this part, we collect the updated synonym vectors of the target word in the j-th sense to generate the target word vector with the j-th sense:

$$\boldsymbol{y}_{w_j} := \boldsymbol{x}_w + \sum_{s\in\mathrm{syn}(w_j)} \boldsymbol{x}_s \tag{13}$$

Moreover, to ensure that each synonym could contribute to the new word vector even if it has no pre-trained vector, we give such a synonym an initial vector randomly, and revise it by the target word vector:

$$\boldsymbol{x}_{s_k} := \lambda\boldsymbol{x}_{s_k} + (1-\lambda)\boldsymbol{x}_w \tag{14}$$

where λ is a weight for unknown neighbor vectors to keep balance of the whole vector space.

3.4 Result

As a result of the procedure, we have sense embeddings of word w

$$\boldsymbol{y}_w = \{\boldsymbol{y}_{w_j}\}_{j=1}^{|\mathrm{sense}_w|}$$

Algorithm 2 NS-sv-Sum

```
Input:
    a pre-trained word embedding $X$;
    the associated vocabulary $\mathcal{V}$;
    the linguistic constraints $LC$;
    the parameter matrix $\boldsymbol{\theta}$;
    the learning rate $\alpha$;
Output:
    a retrofitted sense embedding $Y$;
 1: for each $w \in \mathcal{V}$ do
 2:     find $LC(w) = w, \mathrm{syn}(w), \mathrm{ant}(w)$;
 3:     for each $w_j \in LC(w)$ do
 4:         $\widetilde{w} = \sum_{s \in \mathrm{syn}(w_j)} s$
 5:         for $u \in \{w\} \cup \mathrm{ant}(w)$ do
 6:             $z = g(w, \boldsymbol{\theta})$;
 7:             $p = F_w^{kj}$;
 8:             $g = \alpha(p - z)$;
 9:             $\boldsymbol{v} := \boldsymbol{v} + g\boldsymbol{\theta}_{kj}^u$;
10:             $\boldsymbol{\theta}_{kj}^u := \boldsymbol{\theta}_{kj}^u + g\boldsymbol{x}_{\widetilde{w}}$;
11:         end for
12:         for each $\widetilde{w} \in \mathrm{syn}(w_j)$ do
13:             $\boldsymbol{x}_{\widetilde{w}} := \boldsymbol{x}_{\widetilde{w}} + \boldsymbol{v}$;
14:         end for
15:         $\boldsymbol{y}_{w_j} := \boldsymbol{x}_w + \sum_{\widetilde{w} \in \mathrm{syn}(w_j)} \boldsymbol{x}_{\widetilde{w}}$
16:     end for
17: end for
```

of each word w in the vocabulary $\mathcal{V}$. For each word w, we update the embeddings of the synonyms of w with the j-th sense by the pre-trained word vector of w. Likewise, the antonyms of w are utilized for negative sampling. After traversing of all synonyms, we aggregate these updated synonym embeddings with the pre-trained word embedding $\boldsymbol{x}_w$ to produce the new j-th sense embedding of word w.

The overall procedure of our retrofitting method is sketched as pseudo code in Algorithm 1 for NS-sv and Algorithm 2 for NS-sv-Sum. In Algorithms 1 and 2, the parameter matrix $\boldsymbol{\theta}$ is randomly initialized. Experimental experience tells that initializing the parameter matrix to be the same as the pre-trained vector matrix can generate effective sense retrofitted vectors with only a few iterations. Likewise, it is essential that the parameter matrix $\boldsymbol{\theta}_{kj}^u$ should contribute to $\boldsymbol{v}$ first and then could be updated itself – otherwise we would lose one contribution from $\boldsymbol{\theta}_{kj}^u$ during retrofitting.

4 EXPERIMENTAL SETUP

We measure the effects of our approach on two aspects: semantic relatedness and contextual word similarity. There is a slight difference of dealing with missing words in our experiments from other research. We use the average of all sense vectors to represent the missing word rather than using zero vector or removing the missing word from dataset. And for all models reported in this paper, the same processing method and the same computation method are applied to compare their performance.

4.1 Datasets

We first experiment with three widely used and publicly available pre-trained word vectors for English corpora.

(1) **Word2Vec** [34, 36]: Word2Vec is fast and widely used. In practice, we use the python model[3] from *github.com* [49] to train enwik9[4] by CBOW model and Skip-gram model with Negative Sampling separately in 300 dimensions, where we use context windows of size 5 and 5 negative examples.
(2) **Glove vectors**[5] [39]: The Glove word vector approach integrates the global co-occurrence matrix of word pairs. We use the pre-trained word vectors directly. 6B.50d, 6B.100d, 6B.200d and 6B.300d are trained on Wikipedia 2014 with English Gigaword in vector length of the range 50 to 300 respectively, and 42B.300d is trained on Common Crawl in 300 dimensions.
(3) **FastText vectors**[6] [4]: FastText is an extension of the continuous skip-gram model by summing the n-gram vectors. We use their pre-trained word vector files directly. crawl-300d-2M is trained on Common Crawl including 2 million word vectors, and wiki-300d-1M is trained on Wikipedia 2017, UMBC webbase corpus and *statmt.org* dataset including 1 million word vectors.

For semantic relatedness, we evaluate the quality of the retrofitted embedding spaces on two word similarity benchmarks: **SimLex-999** [23], which comprises 666 noun pairs, 222 verb pairs and 111 adjective pairs; and **SimVerb-3500** [18], which consists of 3500 verb pairs covering all normed verb types of 827 distinct verbs.

For contextual word similarity, we conduct experiments with the **Stanford's Contextual Word Similarities** (SCWS) [24] which includes 2003 word pairs together with human rated scores. Higher scores indicate higher semantic similarity.

As we mentioned above, we generated the synonym sets and antonym sets of the 100,000 most-frequently used words from Wiktionary via the API of *thesaurus.com*[7]. Each word is a leader of its synonym and antonym sets corresponding to its definitions. After inputting them into our program, we are going to mark the leading word as $w\#j$, corresponding to the word w with the j-th sense. It is worth mentioning that we choose *thesaurus.com* as our knowledge base rather than WordNet which considers both semantic relations and lexicon relations. That is because our research is for sense level so that we collect the synonym set and the antonym set according to the different sense of a word directly without caring about more complex relations between words. Beyond that, we decide to use the primary way to gather semantic knowledge.

4.2 Evaluation Measure

Our experiments are all based on intrinsic evaluation for the quality and coherence of vector space. Therefore, we use **Spearman's ρ rank correlation coefficient** [53] between the cosine similarity scores calculated by the retrofitted vectors and the ratings by human to assess how well the relationship between them.

[3]https://github.com/deborausujono/word2vecpy
[4]http://mattmahoney.net/dc/textdata.html
[5]https://nlp.stanford.edu/projects/glove/
[6]https://fasttext.cc/docs/en/english-vectors.html
[7]https://github.com/Manwholikespie/thesaurus

Table 1: MRDS values for test datasets.

Lexicon	Size	$\sigma_{0.01}^{0.5}$	$\sigma_{0.01}^{0.7}$	$\sigma_{0.01}^{0.9}$	$\sigma_{0.05}^{0.5}$	$\sigma_{0.05}^{0.7}$	$\sigma_{0.05}^{0.9}$
SL	999	0.073	0.057	0.032	0.052	0.040	0.023
SV	3500	0.039	0.030	0.017	0.027	0.021	0.012
SCWS	2003	0.051	0.04	0.023	0.036	0.028	0.016

On the other hand, according to Faruqui et al. [15] and Rastogi et al. [42], it is necessary to perform statistical significance tests to the difference between the Spearman Correlations even for comparisons on small evaluation sets. Rastogi et al. [42] introduce $\sigma_{p_0}^{\rho}$ as the *Minimum Required Difference for Significance (MRDS)* which satisfies the the following:

$$(\rho_{AB} < \rho) \wedge \left(|\rho \hat{B}T - \rho \hat{A}T| < \sigma_{p_0}^{\rho}\right) \Rightarrow p\text{-value} > p_0 \tag{15}$$

where A and B are the lists of ratings over the same items, produced by the competitive models and T denotes the gold ratings T. ρ_{AT}, ρ_{BT}, and ρ_{AB} denote the Spearman correlations between $A:T$, $B:T$, and $A:B$, respectively. Then let $\rho_{\hat{A}T}$, $\rho_{\hat{B}T}$, and $\rho_{\hat{A}B}$ be their empirical estimates. This proposition indicates that differences in correlations, if below the MRDS threshold, are not statistically significant. Rastogi et al. [42] also provide the MRDS values for **SimLex-999** (SL) word similarity dataset and here we provide the threshold[8] for **SimVerb-3500** (SV) and **Stanford's Contextual Word Similarities** (SCWS) in Table 1.

4.3 Task 1: Semantic Relatedness

This task is to model the semantic similarity. A higher score indicates the higher semantic similarity. The sense evaluation metrics learned from Reisinger and Mooney [43] compute two kinds of scores, **maximum score** for the evaluation of sense representations and **average score** for the evaluation of word representations:

$$MaxSim = \max_{w_{m_j} \in D_{w_m}, w_{n_k} \in D_{w_n}} \cos\left(\boldsymbol{x}_{w_{m_j}}, \boldsymbol{x}_{w_{n_k}}\right) \tag{16}$$

$$AveSim = \frac{\sum\limits_{w_{m_j} \in D_{w_m}} \sum\limits_{w_{n_j} \in D_{w_n}} \cos\left(\boldsymbol{x}_{w_{m_j}}, \boldsymbol{x}_{w_{n_k}}\right)}{|D_{w_m}| \cdot |D_{w_n}|} \tag{17}$$

where D_{w_m} is the definition set of the word w_m, and D_{w_n} is the definition set of the word w_n. Compared with the original metrics based on unsupervised techniques, we do not need to predetermine the number of sense clusters which should differ from word to word. Thus, instead of the uniform number of clusters, we use the number of senses of each word to average the word similarity.

4.4 Task 2: Contextual Word Similarity

The goal of this task is to measure the semantic relatedness with contextual information which covers the shortage in semantic relatedness task. We also adopt MaxSimC / AvgSimC metrics to compute scores for each word pair [43]. A higher score indicates the higher semantic similarity.

$$MaxSimC = d\left(\hat{\pi}(\boldsymbol{x}_{w_m}), \hat{\pi}(\boldsymbol{x}_{w_n})\right) \tag{18}$$

$$AveSimC = \frac{\sum\limits_{w_{m_j} \in D_{w_m}} \sum\limits_{w_{n_j} \in D_{w_n}} d_{c,w_{m_j}} d_{c',w_{n_k}} d\left(\boldsymbol{x}_{w_{m_j}}, \boldsymbol{x}_{w_{n_k}}\right)}{|D_{w_m}| \cdot |D_{w_n}|} \tag{19}$$

where $d_{c,w_{m_j}} = cos(\boldsymbol{v}(c), \boldsymbol{x}_{w_m})$ is the likelihood of context c belonging to the j-th sense group of the word w_m, and $\hat{\pi}(w_m) = argmax_{w_{m_j} \in D_{w_m}} d_{c,w_{m_j}}$. We select 5 words before and after the target word in the word pairs respectively. Stopwords are removed from the context. There are total 10 word involved in as the context of the target word under the perfect condition. And the context vector of the target word as $\boldsymbol{v}(c)$ are integrated with all context word embeddings. Note that based on our word vector space, the context is not disambiguation, so the solution is the sum of all sense vectors of each context word.

5 RESULTS AND DISCUSSION

For our model, we set a learning rate varying with the number of the synonyms of the target word and the initialized iteration as 1.0. Moreover, the parameter λ to revise the undefined word vector is set to 0.5. In all experiments except the iteration part, we choose $iter = 2$. For statistical significance, we choose $\sigma_{0.05}^{0.9}$, the small threshold value in Table 1.

5.1 Baseline Methods

Tables 2 and 3 show the results of retrofitting the three standard vectors on each benchmark dataset. The second row in each table shows the performance of the baseline methods, i.e., the pre-trained word vectors as defined in each column. If there is only one sense of a word, its maximum score (*MaxSim*) and average score (*AveSim*) will be the same. From Tables 2 and 3, the *MaxSim* and *AveSim* both suggest that the proposed model is robust and useful. The largest improvement is more than 30% on two lexicons' tasks. In contrast with NS-sv-Sum, the synonyms taken as an integration, in general, the first strategy, the synonyms regarded as individual, yields the better results. This can be explained, because the connections among the synonyms are weak. That means the synonyms are the condition of the word sense during our study rather than the constraints of hierarchical semantic relations. Consequently, the first strategy is more logical.

5.2 Comparison with Prior Works

A comparison to prior works is shown in Table 4. The three previous models are: retrofitting-Faruqui[9] [14], GenSense[10] [27], and ER-Specialized[11] [52], all also exploiting external semantic knowledge like synonyms or antonyms for retrofitting word vectors, trained on

[8]https://github.com/se4u/mvlsa provides the way to assign a minimum threshold to a testset.

[9]https://github.com/mfaruqui/retrofitting
[10]https://github.com/y95847frank/GenSense
[11]https://github.com/codogogo/explirefit

Table 2: Spearman's correlation for three word distributed representations, Word2Vec (w2v), Glove (glove) and FastText (FT) on SimLex-999, and the performance comparison with two strategies of retrofitting models, NS-sv and NS-sv-Sum (MaxSim / AveSim) using $\sigma_{0.05}^{0.9}$ as the threshold.

	w2v.cb	w2v.sg	glove.42B	glove.6B	FT.wiki	FT.crawl
baseline	0.230	0.293	0.374	0.371	0.450	0.503
NS-sv	**0.629/0.643**	**0.640/0.635**	**0.671/0.676**	**0.699/0.710**	0.670/**0.703**	0.726/**0.757**
NS-sv-Sum	0.603/0.565	0.581/0.459	0.660/0.599	0.695/0.672	**0.673**/0.608	**0.732**/0.692

Table 3: Spearman's correlation for three word distributed representations, Word2Vec (w2v), Glove (glove) and FastText (FT) on SimVerb-3500, and the performance comparison with two strategies of retrofitting models, NS-sv and NS-sv-Sum (MaxSim / AveSim), using $\sigma_{0.05}^{0.9}$ as the threshold.

	w2v.cb	w2v.sg	glove.42B	glove.6B	FT.wiki	FT.crawl
baseline	0.160	0.184	0.226	0.227	0.357	0.426
NS-sv	**0.565/0.533**	**0.557/0.514**	0.564/0.544	0.593/0.582	**0.591/0.595**	0.624/**0.638**
NS-sv-Sum	0.522/0.471	0.502/0.399	**0.586/0.557**	**0.600/0.584**	0.586/0.538	**0.635**/0.606

Table 4: Spearman's correlation on the SimLex-999 task, prior works and our approach, using $\sigma_{0.05}^{0.9}$ as the threshold.

Corpus	Lexicon	Glove	retrofitting-Faruqui	GenSense	ER-Specialized	NS-sv
Wikipedia	SL	0.265	0.421	0.446/0.417	0.445	**0.658/0.653**
	SV	0.154	0.240	0.289/0.259	0.281	**0.564/0.536**
Twitter	SL	0.122	0.295	0.290/0.244	0.365	**0.626/0.589**
	SV	0.052	0.145	0.160/0.127	0.225	**0.537/0.492**

Table 5: Spearman's correlation on the SCWS task, prior work and our approach (MaxSimC / AvgSimC), computing with the sum of context word embeddings (SCWS-sum) and with the average of context word embeddings (SCWS-avg), using $\sigma_{0.05}^{0.9}$ as the threshold.

	SCWS-sum	**SCWS-avg**
glove.twitter	0.428	0.428
GenSense	0.428/0.322	0.428/0.272
NS-sv	**0.461** /0.335	0.446/0.297

Wikipedia and Twitter using the Glove tool. We run their published code to obtain word vectors. Except for GenSense which needs the weights for their semantic lexicons to discriminate senses of the target word, we applied these models with our new lexicons to decrease the difference. In addition, we kept their default parameter values without any change. Surprisingly, undoubtedly although the prior models still keep their superiority compared with baseline methods, they become somewhat weak when the models only rely on the primary semantic relations. From Table 4, the outcome of this experiment suggests that the quality of our vectors significantly improves compared to each of the three models. The Spearman correlation scores of NS-sv exceed GenSense by more than 20%, which also shows a great performance under word vectors from Wikipedia corpus. And under the word vector from Twitter, NS-sv surpasses ER-Specialized by more than 20%.

5.3 Contextual Word Similarity

Table 5 shows the Spearman correlation of SCWS dataset. With the contextual informationsense embedding models obviously outperform the word embedding model Glove and retrofittig model GenSense. Glove as the baseline method is based on word level so its score is only the similarity between two words in the dataset. The metrics *MaxSimC* and *AveSimC* take the context of the target word into account which are generally used. For the context vector the sum of context word embeddings shows a positive effect on discriminating the distinct senses, since the sum vector could store more information from the context and keep all completed contributions from each word.

5.4 Comparison of Word Vector Length

We tested our model and prior works with different dimensions, using Glove word vectors involving 400K vocabularies. Figure 2 illustrates the different upward trend. The correlation ρ of each model rises continuously from 50 to 300 while the gap among models keeps stable, not varying with more dimensions. We see a similar increment over increasing dimensions except for ER-Specialized which is increasing sharply until arriving 100 dimensions and then reduces the increasing rate. In other words, the increasing of dimensions has little effect on the model itself. These changes affect the accuracy of word vectors and our retrofitting model improves the vectors based on the original accuracy of vectors. This reflects the robustness and stability of our neural network model to a level comparable to graph-based models. The vector length in 300 is the most commonly used because it yields a good trade-off between high accuracy and acceptable data size. It is worth noting that even with

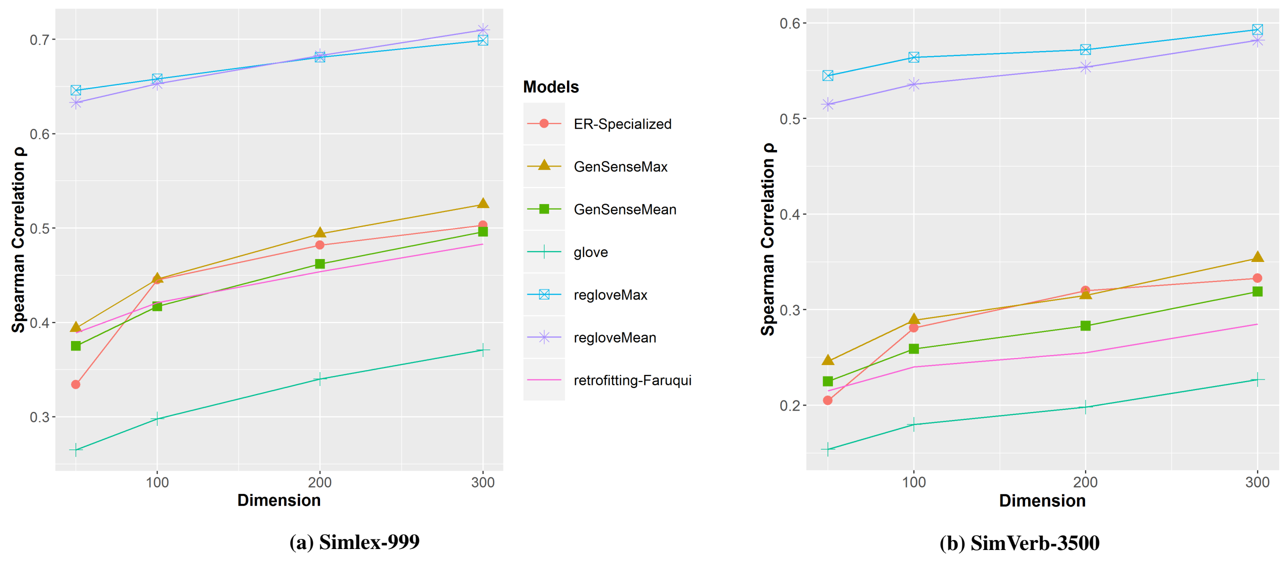

Figure 2: Spearman's correlation on word similarity tasks in different vector dimensions.

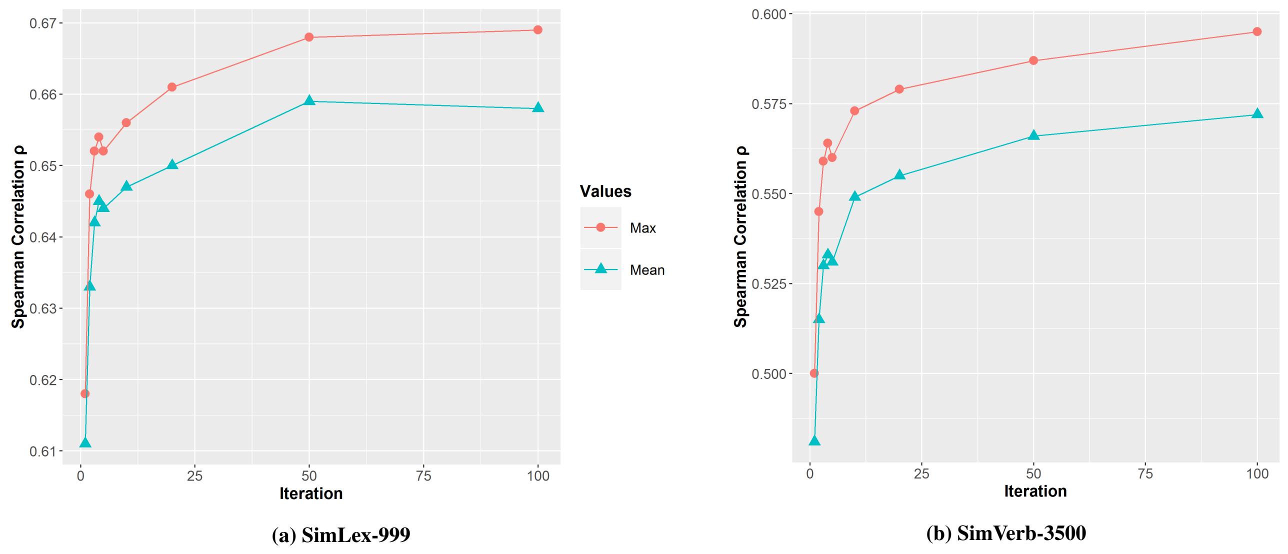

Figure 3: The influence of number of iterations on word similarity tasks.

50 dimensions, the sense vector generated by our model can still be highly accurate which helps to easily apply the low dimensional vectors for applications instead of training the large and complex word vectors with much consumption. Thus, the retrofitting work is reasonable and meaningful in practice.

5.5 The Influence of Number of Iterations

We conduct an experiment to test the influence of the number of iterations and observe how much benefit we can get from more iterations. Based on the outcomes in Figure 3, the correlation score grows in fluctuations and peaks at iteration 4. There is a small decrease of score after the 4th iteration. According to Early Stopping [57], we would like our model to produce a good performance with less cost and avoid overfitting. So we set a cycle to test the loss. However, the number of iterations from 4 to 50 merely show little improvement (less than 2%). After the 50th iteration, the accuracy begins to drop again or the growth rate is reducing. As a result, we suggest to select $iter = 4$ as the default iteration for our model. This way we achieve a reasonable trade-off between performance and running time.

5.6 Running Time

We executed our code on a computer with 7.7 GB main memory and an Intel Core i5-4570. We do not consider the time to read from or write to files, since these processes are not related to our main code. It took 19.8 seconds to retrofit 1 GB of word vectors. This was significantly faster than the other network-based method, with ER-Specialized taking more than four hours under the same conditions. The graph-based retrofitting-Faruqui took 9.6 seconds when it only applied synonym constraints, and GenSense took 16.2 seconds when it used the preset weights of each synonym and antonym. As these are just single performance measurements, we have to take them with a grain of salt [26], but these observations are suggesting that our retrofitting approach is quite competitive in terms of efficiency.

6 CONCLUSION

We presented a technique to retrofit word vectors from word level to more fine-grained level of word sense by two strategies named NS-sv and NS-sv-Sum in this paper. This technique employs semantic knowledge to improve the performance of the pre-trained word vectors and partitions the meaning of conflated words into multiple senses. This is a post-processing approach avoiding retraining a large corpus and can be applied on any pre-trained word vectors. Our model only uses the primary thesaurus which is easy to access. Furthermore, during the retrofitting procedure, we do not need any weights for semantic knowledge assigned by humans. There are merely three parameters required, namely learning rate, the number of iterations, and a weight for unknown neighbor vectors, in addition to pre-trained word vectors and external semantic knowledge.

Our model consists of four parts: (1) We provide the synonym and antonym sets of the most commonly used words in Wiktionary and use them as training examples for our neural network model rather than as external constraints in linear models; (2) our model makes use of negative sampling to reduce the cost and uncertainty of the hidden layer, which differs from previous neural network models; (3) our model significantly reduces the number of parameters and, based on experiments, the default parameters do typically not have to be modified even if the input and the environment change; (4) we take advantage of the semantic knowledge from external resources to construct sense representations. Our experiments have provided evidence for the effectiveness of our model on word similarity tasks where it outperforms previous work. Also in terms of runtime it is superior to the second best approach, giving a better performance in significantly less training time.

Future research could apply generalized sense representations in downstream NLP processing applications and carry out an extrinsic evaluation of our approach.

REFERENCES

[1] Ben Athiwaratkun and Andrew Gordon Wilson. 2017. Multimodal word distributions. *arXiv preprint arXiv:1704.08424* (2017).

[2] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL '98, August 10-14, 1998, Université de Montréal, Montréal, Quebec, Canada. Proceedings of the Conference.* 86–90. http://aclweb.org/anthology/P/P98/P98-1013.pdf

[3] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia-A crystallization point for the Web of Data. *Web Semantics: science, services and agents on the world wide web* 7, 3 (2009), 154–165.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *TACL* 5 (2017), 135–146. https://transacl.org/ojs/index.php/tacl/article/view/999

[5] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, 1247–1250.

[6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[7] Roger Burrowes Bradford. 2010. Machine translation using vector space representations. (July 27 2010). US Patent 7,765,098.

[8] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal Distributional Semantics. *J. Artif. Intell. Res.* 49 (2014), 1–47. DOI:http://dx.doi.org/10.1613/jair.4135

[9] José Camacho-Collados and Mohammad Taher Pilehvar. 2018. From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. *CoRR* abs/1805.04032 (2018). http://arxiv.org/abs/1805.04032

[10] 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. *Artificial Intelligence* 240 (2016), 36–64.

[11] Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-Relational Latent Semantic Analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. 1602–1612. http://aclweb.org/anthology/D/D13/D13-1167.pdf

[12] 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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*. 15–20. http://aclweb.org/anthology/P/P15/P15-2003.pdf

[13] Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. 897–906.

[14] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*. 1606–1615. http://aclweb.org/anthology/N/N15/N15-1184.pdf

[15] Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems With Evaluation of Word Embeddings Using Word Similarity Tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*. Association for Computational Linguistics, Berlin, Germany, 30–35. DOI:http://dx.doi.org/10.18653/v1/W16-2506

[16] 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 (2002), 116–131. DOI:http://dx.doi.org/10.1145/503104.503110

[17] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 758–764.

[18] Daniela Gerz, Ivan Vulic, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. 2173–2182. http://aclweb.org/anthology/D/D16/D16-1235.pdf

[19] Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. 2015. Random Walks and Neural Network Language Models on Knowledge Bases. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*. 1434–1439. http://aclweb.org/anthology/N/N15/N15-1165.pdf

[20] Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*. 427–431. https://aclanthology.info/papers/E17-2068/e17-2068

[21] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 297–304.

[22] Zellig S Harris. 1954. Distributional structure. *Word* 10, 2-3 (1954), 146–162.

[23] Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. *Computational Linguistics* 41, 4 (2015), 665–695. DOI:http://dx.doi.org/10.1162/COLI_a_00237

[24] 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. Association for Computational Linguistics, 873–882.

[25] Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing Word Embeddings for Similarity or Relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. 2044–2048. http://aclweb.org/anthology/D/D15/D15-1242.pdf

[26] Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. 2017. The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowl. Inf. Syst.* 52, 2 (2017), 341–378.

[27] Yang-Yin Lee, Ting-Yu Yen, Hen-Hsen Huang, Yow-Ting Shiue, and Hsin-Hsi Chen. 2018. GenSense: A Generalized Sense Retrofitting Model. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*. 1662–1671. https://aclanthology.info/papers/C18-1141/c18-1141

[28] Benjamin J. Lengerich, Andrew L. Maas, and Christopher Potts. 2018. Retrofitting Distributional Embeddings to Knowledge Graphs with Functional Relations. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*. 2423–2436. https://aclanthology.info/papers/C18-1205/c18-1205

[29] Jiwei Li and Dan Jurafsky. 2015. Do Multi-Sense Embeddings Improve Natural Language Understanding? (2015), 1722–1732. http://aclweb.org/anthology/D/D15/D15-1200.pdf

[30] Wei Li and Andrew McCallum. 2005. Semi-Supervised Sequence Modeling with Syntactic Topic Models. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*. 813–818. http://www.aaai.org/Library/AAAI/2005/aaai05-128.php

[31] Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning Semantic Word Embeddings based on Ordinal Knowledge Constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. 1501–1511. http://aclweb.org/anthology/P/P15/P15-1145.pdf

[32] 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, January 25-30, 2015, Austin, Texas, USA*. 2418–2424. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9314

[33] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*. 142–150. http://www.aclweb.org/anthology/P11-1015

[34] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). http://arxiv.org/abs/1301.3781

[35] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*

[36] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality

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

[38] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. *CoRR* abs/1504.06654 (2015). http://arxiv.org/abs/1504.06654

[39] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 1532–1543. http://aclweb.org/anthology/D/D14/D14-1162.pdf

[40] Mohammad Taher Pilehvar and Nigel Collier. 2016. De-Conflated Semantic Representations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. 1680–1690. http://aclweb.org/anthology/D/D16/D16-1174.pdf

[41] Lin Qiu, Kewei Tu, and Yong Yu. 2016. Context-Dependent Sense Embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. 183–191. http://aclweb.org/anthology/D/D16/D16-1018.pdf

[42] Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation Learning via Generalized CCA. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, 556–566. DOI:http://dx.doi.org/10.3115/v1/N15-1058

[43] Joseph Reisinger and Raymond J Mooney. 2010. 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*. Association for Computational Linguistics, 109–117.

[44] Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

[45] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (1975), 613–620.

[46] Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational linguistics* 24, 1 (1998), 97–123.

[47] Yow-Ting Shiue and Wei-Yun Ma. 2017. Improving word and sense embedding with hierarchical semantic relations. In *2017 International Conference on Asian Language Processing, IALP 2017, Singapore, December 5-7, 2017*. 350–353. DOI:http://dx.doi.org/10.1109/IALP.2017.8300615

[48] Victor Hugo Andrade Soares, Ricardo J. G. B. Campello, Seyednaser Nourashrafeddin, Evangelos Milios, and Murilo Coelho Naldi. 2019. Combining semantic and term frequency similarities for text clustering. *Knowledge and Information Systems* (2019). DOI:http://dx.doi.org/10.1007/s10115-018-1278-7

[49] Debora Sujono. 2015. word2vecpy. https://github.com/deborausujono/word2vecpy. (2015).

[50] Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2011. Latent vector weighting for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1012–1022.

[51] Florentina Vasilescu, Philippe Langlais, and Guy Lapalme. 2004. Evaluating Variants of the Lesk Approach for Disambiguating Words. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*. http://www.lrec-conf.org/proceedings/lrec2004/pdf/219.pdf

[52] Ivan Vulic and Goran Glavas. 2018. Explicit Retrofitting of Distributional Word Vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. 34–45. https://aclanthology.info/papers/P18-1004/p18-1004

[53] Arnold D Well and Jerome L Myers. 2003. *Research design & statistical analysis*. Psychology Press.

[54] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789* (2016).

[55] Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A General Framework for Incorporating Knowledge into Word Representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. 1219–1228. DOI:http://dx.doi.org/10.1145/2661829.2662038

[56] Xuefeng Yang and Kezhi Mao. 2016. Learning multi-prototype word embedding from single-prototype word embedding with integrated knowledge. *Expert Syst. Appl.* 56 (2016), 291–299. DOI:http://dx.doi.org/10.1016/j.eswa.2016.03.013

[57] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. 2007. On early stopping in gradient descent learning. *Constructive Approximation* 26, 2 (2007), 289–315.

[58] Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xue-Jie Zhang. 2017. Refining Word Embeddings for Sentiment Analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. 534–539. https://aclanthology.info/papers/D17-1056/d17-1056

[59] Mo Yu and Mark Dredze. 2014. Improving Lexical Embeddings with Semantic Knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*. 545–550. http://aclweb.org/anthology/P/P14/P14-2089.pdf

[60] Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning Continuous Word Embedding with Metadata for Question Retrieval in Community Question Answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. 250–259. http://aclweb.org/anthology/P/P15/P15-1025.pdf