Leveraging Web Semantic Knowledge in Word Representation Learning

Haoyan Liu,[1*] **Lei Fang,**[2] **Jian-Guang Lou,**[2] **Zhoujun Li**[1]
[1]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[2]Microsoft Research, Beijing, China
{haoyan.liu, lizj}@buaa.edu.cn; {leifa, jlou}@microsoft.com

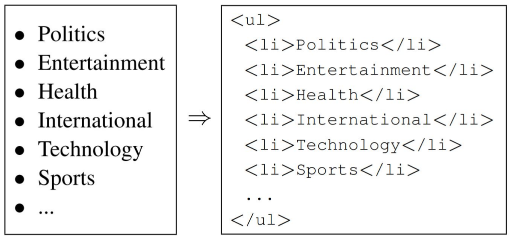
Abstract

Much recent work focuses on leveraging semantic lexicons like WordNet to enhance word representation learning (WRL) and achieves promising performance on many NLP tasks. However, most existing methods might have limitations because they require high-quality, manually created, semantic lexicons or linguistic structures. In this paper, we propose to leverage semantic knowledge automatically mined from web structured data to enhance WRL. We first construct a semantic similarity graph, which is referred as semantic knowledge, based on a large collection of semantic lists extracted from the web using several pre-defined HTML tag patterns. Then we introduce an efficient joint word representation learning model to capture semantics from both semantic knowledge and text corpora. Compared with recent work on improving WRL with semantic resources, our approach is more general, and can be easily scaled with no additional effort. Extensive experimental results show that our approach outperforms the state-of-the-art methods on word similarity, word sense disambiguation, text classification and textual similarity tasks.

1 Introduction

Distributed word representations boost performance of many NLP applications mainly because they are capable of capturing semantic regularities from a collection of text sequences. Much research work tries to enhance word representation learning (WRL) from the semantic perspective by leveraging semantic lexicons. Semantic lexicons can be considered as a collection of lists, in which each list consists of semantically related words. Some existing work pulls the vectors of synonyms close by either a post-processing model (Faruqui et al. 2015) or a joint representation learning model with the distances between synonyms as regularizers (Kiela, Hill, and Clark 2015; Bollegala et al. 2016). More recently, many manually well-designed semantic relations or linguist structures, such as synonyms and antonyms (Mrkšić et al. 2017; Glavaš and Vulić 2018), concept convergence and word divergence (Liu et al. 2018), have been used to enhance the semantics of words.

However, it should be noted that most existing models might have limitations because they require high-quality, manually created, semantic lexicons or linguistic structures. Even for high-quality semantic resources like WordNet[1], the coverage might be quite limited. Taking English WordNet as an example, it only contains 155K words organized in 176K synsets, which is rather small compared to the large vocabulary size on the training data. Vulić et al. (2018) and Glavaš and Vulić (2018) partially solve this problem by first designing a mapping function that learns the specialization process for seen words, and then applying the learned function to unseen words in semantic lexicons. Unfortunately, their approaches still depend on the linguistic constraints derived from manually created resources. Therefore, we shall leverage semantic resources that can be automatically constructed with a relatively high coverage on the vocabulary.



- Politics
- Entertainment
- Health
- International
- Technology
- Sports
- ...

⇒

```
<ul>
  <li>Politics</li>
  <li>Entertainment</li>
  <li>Health</li>
  <li>International</li>
  <li>Technology</li>
  <li>Sports</li>
  ...
</ul>
```

Figure 1: News categories with HTML structures.[2]

There is a considerable amount of structured data on the web, such as tables, select options, drop down lists, etc. Given a web table, entries in the same column or row are usually highly semantically related. This inspires the idea that it is promising to automatically construct the semantic resources based on those semantically related entries. Figure 1 shows an example of news categories with their corresponding HTML structures. It can be seen that *Politics*, *Entertainment*, *Health* are semantically related, because all of them can be treated as news categories. In the HTML DOM tree, *Politics*, *Entertainment*, *Health* share the same HTML structure, which means that they can be easily extracted using HTML tag patterns. Moreover, we find that semantic information in structured data could be an enhancement or complement for text corpora. Consider the entries

*This work was done when the first author was an intern at Microsoft Research Asia.

[1]https://wordnet.princeton.edu/

[2]The categories are from https://abcnews.go.com/.

in news categories again, in text sequences, they hardly appear with similar contexts, but they might frequently appear in the navigation bars on news websites.

Therefore, we propose to use some pre-defined HTML tag patterns to extract semantic lists from web data in a large scale. However, the extracted semantic lists cannot be directly used by existing methods, because they contain much noise, and some of the lists are highly redundant. To address this issue, we design a similarity function that measures the semantic relatedness between each co-occurred word pair. After that, we build a semantic similarity graph, in which words are represented as vertices and each edge indicates the similarity score between the corresponding two vertices. This semantic similarity graph is considered as web semantic knowledge to further enhance WRL.

In this paper, we propose WESEK, **W**ord **E**mbedding with web **SE**mantic **K**nowledge, to capture semantics from both the text and web semantic knowledge. The intuition behind WESEK is that, given two words, the similarity of learned representations shall be high if they are connected in the semantic similarity graph. Our major contributions are: 1) we build high-quality semantic knowledge from web structured data, which can be easily scaled with no additional human effort; 2) we propose a novel joint representation learning model that is capable of capturing semantics from both the text and semantic knowledge. Experimental results show that WESEK outperforms state-of-the-art word embeddings on multiple popular NLP tasks like word similarity, word sense disambiguation, text classifications and textual similarity, which demonstrate that leveraging web semantic knowledge improves WRL and that WESEK is capable of encoding web semantic knowledge effectively.

2 Related Work

Our work is related to 1) word representation specialization or retrofitting using semantic lexicons or linguistic structures, and 2) semantic knowledge construction from the web.

2.1 Word Representation Specialization

Most specialization models make use of external resources like WordNet or the Paraphrase Database[3] (PPDB) to derive semantic lexicons or linguistic structures. Generally, they fall into two categories: post-processing and joint learning.

Post-processing models. The inputs of post-processing models are pre-trained word embeddings, which means that embeddings will not be retrained during specialization. Faruqui et al. (2015) retrofit embeddings with an efficient iterative updating method to reduce the distances between synonyms derived from WordNet. Vulić et al. (2018) and Glavaš and Vulić (2018) propose to learn specialization functions of seen words in semantic lexicons and propagate it to unseen words. Much research work (Mrkšić et al. 2016; 2017; Glavaš and Vulić 2018) utilizes antonyms to further differentiate the dissimilar words in addition to pulling the representation of synonyms words close. Some other linguistic structures can also be utilized, for instance, Vulić et al. (2017) exploit morphological synonyms to pull the inflectional forms of the same word closer together and push derivational antonyms further apart.

[3]http://paraphrase.org

Joint learning models. Many joint learning models introduce semantic lexicons or linguistic structures as additional constraints to the representation learning models (Liu et al. 2015; Ono, Miwa, and Sasaki 2015; Nguyen et al. 2017). Yu and Dredze (2014) integrate word2vec (Mikolov et al. 2013) with synonym constraints that the distances between synonym representations shall be close. Liu et al. (2018) introduce the semantic constraints of concept convergence and word divergence derived from hypernym-hyponym relations in WordNet to word2vec. Bollegala et al. (2016) extend GloVe (Pennington, Socher, and Manning 2014) by adding constraints of various semantic relationships such as synonyms, antonyms, hypernyms and hyponyms. Osborne, Narayan, and Cohen (2016) propose to find a projection of the two views (word embeddings and semantic knowledge) in a shared space such that the correlation between the two views is maximized with minimal redundancy. For other joint learning models, Kiela, Hill, and Clark (2015) treat semantically related words as alternative contexts by adding them to the original context window in text corpora; Niu et al. (2017) utilize word sememes, which are the minimum semantic units of word meanings, to improve WRL with an attention scheme.

2.2 Semantic Knowledge Construction

Both **post-processing** and **joint learning** models require high-quality, manually constructed, semantic lexicons or linguistic structures (Nguyen et al. 2017; Liu et al. 2018; Mrkšić et al. 2016; Glavaš and Vulić 2018). It should be noted that structured data on the web contains rich semantic information, which has the advantage that it can be easily extracted (Zhang et al. 2013). There is much related work on mining semantic knowledge automatically from web data (Pasca 2004; Zhang et al. 2009; Shi et al. 2010; Wu et al. 2012); which generally uses Hearst patterns (Hearst 1992) to discover coordinative words in the text, and HTML tag patterns to extract words sharing the same HTML structure. Shi et al. (2010) propose using both textual context and HTML tag patterns to extract semantic lists from web data; and Zhang et al. (2009) utilize topic models to try to obtain high-quality semantic classes from raw semantic lists extracted from the web. In this paper, we first extract a large collection of semantic lists with several pre-defined HTML tag patterns, and then construct a semantic similarity graph with high quality.

3 Web Semantic Knowledge Extraction

There are two major steps to construct semantic knowledge from web structured data: (1) semantic lists extraction, and (2) similarity graph construction. The similarity graph, which is also referred as semantic knowledge, is further utilized to specialize word representation learning.

3.1 Semantic Lists Extraction

We assume that entries in the same list or table on the web are semantically related. Similar to previous work (Shi et al.

2010), we use HTML tag patterns to extract semantic lists - an extraction approach that can be easily scaled with more data. A straightforward way to perform the approach is to use HTML tags like *table* or *list* to extract semantic lists. In addition, we also use HTML tag repeat patterns. For instance, after parsing the HTML page to a DOM tree, text nodes on the same level are extracted as a semantic list if their ancestors upward to the root node are the same.

Type	Patterns
table	<table> <tr> <td> T </td> ... </tr> ... <tr> <td> T </td> ... </tr> </table>
list	<ul> <li> T </li> ... <li> T </li> </ul>
	<ol> <li> T </li> ... <li> T </li> </ol>
	<select> <option> T ... <option> T </select>
Other HTML tag repeat patterns	

Table 1: HTML patterns for semantic list extraction.

Table 1 shows the HTML tag patterns utilized in this paper, where T is an entry in the extracted semantic list.

3.2 Similarity Graph Construction

However, semantic lists extracted from the web cannot be a direct replacement for the semantic lexicons leveraged in previous work, because they may contain too much noise or be highly redundant. According to our estimation on the sampled 10 million semantic lists, at least 20% are near duplicate or contained by others[4]. Moreover, for some domains, the extracted lists are not quite semantically related but have relatively high frequencies. As most existing work requires high-quality semantic lexicons, directly introducing the extracted semantic lists might harm performance.

To address this issue, we design a similarity function extended from Zhang et al. (2009) and Shi et al. (2010) for each co-occurred word pair as:

$$s_{ab} = (\sum_{i=1}^{m} \log(1 + n_{i,ab}))\sqrt{\log(1 + \frac{N}{N_a})\log(1 + \frac{N}{N_b})}, \quad (1)$$

where $n_{i,ab}$ is the number of times that words a and b co-occurred in the same list in domain i, and there are m domains in total; N is the total number of semantic lists extracted from the web; N_a, N_b are the numbers of semantic lists that contain words a and b, respectively. To make it clear, the right-side of this function (the radical part) considers the frequencies of words a and b in the whole corpora, which is similar to the inverse document frequency (IDF). For the left-side, a pair of words should have a high score if they occurred in multiple websites with relatively high co-occurrences. It can be seen that the similarity scores can be computed in parallel; and the overall computation is quite efficient, as similarity scores are computed only for word pairs that co-occur. Till now, we obtain the similarity graph constructed from the web semantic lists.

To assure that the semantic resource has high quality, we prune the graph by removing edges with low similarity scores. Given a word a, let $\{b_1, b_2, ..., b_K\}$ denote the neighbors of a in decreasing order of similarity score. We remove the long tailed neighbors to obtain the top k semantically related neighbors by

$$\sum_{i=1}^{k} s_{ab_i} \leq \mu \sum_{i=1}^{K} s_{ab_i} < \sum_{i=1}^{k+1} s_{ab_i}, \quad (2)$$

where μ is the threshold between 0 and 1, s_{ab_i} is the similarity of a and b_i. The pruned similarity graph is considered as the high-quality semantic knowledge to improve WRL later.

4 WRL with Semantic Knowledge

We leverage our scalable semantic knowledge pipeline for WRL by extending the word2vec models proposed by Mikolov et al. (2013). While the word2vec models include continuous bag-of-word (CBOW) and skip-gram (SG), due to limited space, we will only discuss details of WESEK based on SG. It can be easily demonstrated that our approach also applies to CBOW.

4.1 Skip-gram and Negative Sampling

Skip-gram is an efficient method to learn high-quality word representations from the text. The objective of SG is to learn word representations that can accurately predict the surrounding words of a given word. Given a sequence of words $w_1, w_2, \ldots, w_T$, SG aims to maximize the log probability:

$$\mathcal{O}_{\mathcal{C}} = \sum_{t=1}^{T} \sum_{-d \leq j \leq d, j \neq 0} \log p(w_{t+j}|w_t), \quad (3)$$

where d is the local window size - words in this window are treated as contexts of the target word w_t. $p(w_{t+j}|w_t)$ is the predictive probability of context word w_{t+j} conditioned on the target word w_t, defined by the softmax function:

$$p(w_c|w_t) = \frac{\exp(v_c^\top u_t)}{\sum_{k=1}^{|V|} \exp(v_k^\top u_t)}, \quad (4)$$

where u_t and v_c are the "input" and "output" vector representations of words w_t and w_c, respectively. $|V|$ is the number of words in the vocabulary. The input embedding usually stands for the original word embedding while output embedding denotes the word in the context. Intuitively, words with similar contexts shall have similar representations.

Generally, the vocabulary size is very large. It is extremely expensive to calculate the softmax function over the whole vocabulary. One computationally efficient approximation is the negative sampling approach. Instead of making predictions on the entire vocabulary space, negative sampling makes a binary prediction of whether a word is present or absent in the contexts for a target word. Formally, for a target word w_t at position t, let all context words within the window size be positive examples, and randomly sample negative instances from the vocabulary. The objective

[4]We use MinHash to detect near duplicate semantic lists. To estimate the percentage of "contain" relation, we sort entries in each list and sort all the lists in lexicographic order. Then all the lists are divided into small sets, where each set contains 50K consecutive lists. For each set of lists, we perform a pairwise check to see if one list is a subset of the other.

of negative sampling is to maximize the following negative log-likelihood:

$$\mathcal{L}_{\mathcal{C}} = \sum_{t=1}^{T} \sum_{w_c \in \mathcal{C}_t} [\log \sigma(v_c^\top u_t) + \sum_{w_n \in \mathcal{N}_{t,c}} \log \sigma(-v_c^\top u_n)], \quad (5)$$

where $\sigma(x) = 1/(1+\exp(-x))$, $\mathcal{C}_t$ is the context set for target word w_t, and $\mathcal{N}_{t,c}$ is a set of negative samples drawn from the noise distribution $P_n(w)$ for context word w_c. Following Mikolov et al. (2013), we set $P_n(w) \propto F_w^{3/4}$, where F_w is the frequency of word w.

4.2 WESEK: Word Embedding with Semantic Knowledge

In this section, we present the details on how semantic knowledge is incorporated in WESEK. The intuition behind WESEK is that, given two words, the similarity of learned representations shall be high if they are connected in the semantic similarity graph.

Problem formulation. Let s_{ij} denote the similarity score between w_i and w_j in the similarity graph. Given a target word w_t, $\mathcal{S}_t$ is the set of all corresponding neighbors in the pruned similarity graph. For each neighbor word w_l of w_t, we define their relevance score as:

$$\hat{r}(w_l|w_t) = \frac{s_{tl}}{\sum_{w_k \in \mathcal{S}_t} s_{tk}}. \quad (6)$$

From the view of word representations, we define the semantic relevance score between w_t and w_l as:

$$r(w_l|w_t) = \sigma(u_l^\top u_t), \quad (7)$$

where $\sigma(x)$ is the sigmoid function. Since both relevance scores are defined given target word w_t, we simplify $\hat{r}(w_l|w_t)$ and $r(w_l|w_t)$ as $\hat{r}_{tl}$ and r_{tl}, respectively.

Our goal is to maximize the similarity of the learned word representations if the words are connected in the semantic similarity graph. Given the word w_t and its neighbor set $\mathcal{S}_t$, we aim to maximize the following objective:

$$\mathcal{O}_{\mathcal{S}_t} = \log \prod_{w_l \in \mathcal{S}_t} r_{tl}^{\hat{r}_{tl}} = \sum_{w_l \in \mathcal{S}_t} \hat{r}_{tl} \log \sigma(u_l^\top u_t). \quad (8)$$

It can be seen that the $\mathcal{O}_{\mathcal{S}_t}$ combines both relevance scores from the text and semantic knowledge, and it can be proved that maximizing the objective is consistent with our goal.

Instead of optimizing the above objective directly, we employ the negative sampling approach in a similar way as (Mikolov et al. 2013). For the target word w_t and its neighbor w_l, we draw a set of negative samples (words that are not connected with w_t in the graph), denoted as $\mathcal{N}_{t,l}$, from the noise distribution $P_n(w)$. It is evident that for each word $w_n \in \mathcal{N}_{t,l}$, its relevance score with w_t, denoted by r_{tn}, should be low. Therefore, we define $1 - r_{tn}$ as the irrelevance score and rewrite the objective $\mathcal{O}_{\mathcal{S}_t}$ as:

$$\begin{aligned} \mathcal{O}_{\mathcal{S}_t} &= \sum_{w_l \in \mathcal{S}_t} \hat{r}_{tl} [\log r_{tl} + \sum_{w_n \in \mathcal{N}_{t,l}} \log(1 - r_{tn})] \\ &= \sum_{w_l \in \mathcal{S}_t} \hat{r}_{tl} [\log \sigma(u_l^\top u_t) + \sum_{w_n \in \mathcal{N}_{t,l}} \log \sigma(-u_n^\top u_t)]. \end{aligned} \quad (9)$$

To jointly learn representations from the text and semantic knowledge, an easy way is to introduce the objective to the SG model. However, it is still time-consuming to update the representations for the target word and all its neighbors. We propose *positive sampling*, i.e., instead of updating the representations for all the neighbors, we only update a sampled subset of neighbors, which can be regarded as *positive* samples. Given a target word w_t, considering the relevance score in semantic knowledge, we draw positive samples from the distribution $P_p(t) \propto \hat{r}_{tl}$ to obtain the positive sample set $\mathcal{P}_t$. Therefore, the objective becomes:

$$\mathcal{O}_{\mathcal{S}_t} = \sum_{w_l \in \mathcal{P}_t} [\log \sigma(u_l^\top u_t) + \sum_{w_n \in \mathcal{N}_{t,l}} \log \sigma(-u_n^\top u_t)]. \quad (10)$$

Then, for every target word w_t in training data, the goal is to maximize the following objective function:

$$\mathcal{L}_{\mathcal{S}} = \sum_{t=1}^{T} \sum_{w_l \in \mathcal{P}_t} [\log \sigma(u_l^\top u_t) + \sum_{w_n \in \mathcal{N}_{t,l}} \log \sigma(-u_n^\top u_t)]. \quad (11)$$

By integrating it with the original SG model, the joint objective is then to maximize:

$$\mathcal{L} = \mathcal{L}_{\mathcal{C}} + \lambda \mathcal{L}_{\mathcal{S}}, \quad (12)$$

where λ balances the weight between the text and semantic knowledge. After that, we obtain the resulting word embedding with semantic knowledge (WESEK).

5 Experiments

To demonstrate the effectiveness of incorporating semantic knowledge for WRL, we evaluate WESEK over several popular tasks, namely word similarity, word sense disambiguation, text classification, and textual similarity[5].

5.1 Experiments Setup

To make sure that comparisons are fair, we train all embeddings on the English Wikipedia dump[6]. Words with a frequency below 5 are filtered out. The training data has around 1.2 billion tokens with a vocabulary size of 2.9 million.

Semantic knowledge construction. We extract a large collection of semantic lists from the Common Crawl data[7] using the patterns defined in Table 1 and filter out entries that do not exist in the vocabulary of the training data. Lists with the number of entries lower than 3 are removed, and we only select words with a frequency above 5. After that, we construct the semantic similarity graph as described in Section 3. We prune the graph by removing edges with low similarity scores using Equation 2, with $\mu = 0.2$, i.e., for each word, we select top weighted neighbors that their accumulated sum of similarity score contributes to at least 20% of the total sum. The performance is rather stable when μ is between 0.1-0.5 and is slightly decreased when it is above 0.5. The resulting pruned similarity graph - semantic knowledge - has 1.6 million nodes and 8.3 million edges (which suggests the semantic knowledge has a relatively high coverage of the vocabulary).

[5]Code and data to reproduce the results are available at https://github.com/haoyanliu/wesek.

[6]http://dumps.wikimedia.org/enwiki/

[7]http://commoncrawl.org/

	MC30	MEN-TR	MTurk771	RG65	RW	WS353-ALL	WS353-REL	WS353-SIM	[AVG]
GloVe$_{\text{pre}}$	70.3	73.8	65.0	76.6	41.2	60.5	57.3	66.4	63.9
word2vec$_{\text{pre}}$	78.8	77.1	67.1	75.0	53.4	69.2	62.2	77.7	70.1
fastText$_{\text{pre}}$	83.6	79.1	71.0	84.5	52.3	70.8	65.0	81.0	73.4
Retro$_{\text{WN}}$	82.1	73.3	66.0	<u>83.6</u>	32.2	63.7	53.1	75.1	66.1
Retro$_{\text{PPDB}}$	81.6	75.5	<u>70.4</u>	81.5	45.1	68.8	61.2	77.3	70.2
CF	77.2	67.7	60.8	76.7	41.0	57.8	50.2	65.3	62.1
AR	<u>85.2</u>	74.6	65.1	80.7	44.9	68.5	60.9	76.4	69.5
PostSpec$_{\text{CF}}$	77.2	66.3	59.7	76.7	39.0	56.9	49.2	63.9	61.1
PostSpec$_{\text{AR}}$	79.2	71.1	63.7	77.8	42.4	65.5	57.3	73.8	66.4
SENSE	82.2	74.2	65.0	76.4	40.0	70.7	63.4	77.7	68.7
GloVe	75.3	67.0	62.9	80.4	30.4	56.3	49.2	67.6	61.1
word2vec	81.3	74.7	65.4	81.8	44.6	70.8	64.1	77.8	70.1
fastText	80.1	<u>76.2</u>	65.7	81.4	<u>47.8</u>	71.1	64.6	77.2	70.5
WESEK	**82.7**	76.1	**67.3**	**83.0**	46.6	<u>**72.5**</u>	<u>**66.2**</u>	<u>**79.5**</u>	<u>**71.7**</u>

Table 2: Spearman's ρ coefficient $\times 100$ on word similarity tasks. [AVG] means the average over all tasks. The numbers in **bold** mean that WESEK outperforms *word2vec*, *GloVe*, and *fastText*. The <u>underlined</u> numbers denote the best performance achieved on English Wikipedia dump.

Baselines and parameter settings. We compare WESEK with *word2vec*, *fastText* (Bojanowski et al. 2017), *GloVe*, and some word representation specialization methods that leverage semantic lexicons or linguistic structures. The default size of the utilized word vectors is 300. For *word2vec*, we use the skip-gram model with negative sampling; set both context window size and the number of negative samples as 10, learning rate as 0.025; and run the algorithm for 3 iterations. The *word2vec* embedding is considered as the input baseline embedding for other representation specialization methods. *GloVe* and *fastText* also use default parameter settings. For other baselines, parameter settings are:

- *retrofitting* (*Retro*), proposed by Faruqui et al. (2015), is a post-processing approach to reduce the distances between synonyms iteratively. We report *retrofitting* that uses the *word2vec* baseline embedding with lexicons derived from WordNet and PPDB, denoted by *Retro*$_{\text{WN}}$ and *Retro*$_{\text{PPDB}}$, respectively.
- *counter-fitting* (*CF*) and *attract-repel* (*AR*), proposed by Mrkšić et al. (2016) and Mrkšić et al. (2017), introduce antonym and synonymy constraints for WRL. Mrkšić et al. (2016) show that with antonyms from WordNet and PPDB, synonyms from PPDB as semantic lexicons, *CF* achieves the best performance; we report *CF* under this setting, and *AR* under its default setting, both with the *word2vec* baseline embedding as inputs.
- *Post-Specialisation* (*PostSpec*), proposed by Vulić et al. (2018), uses a deep neural network to learn specialization functions of seen words in semantic lexicons, and applies the learned functions to unseen words. We use the *word2vec* baseline embedding as inputs to train two *PostSpec* models: *PostSpec*$_{CF}$ with *counter-fitting* as the output, *PostSpec*$_{AR}$ with *attract-repel* as the output. Then we obtain the specialized embeddings for the whole vocabulary.
- *SENSE*, proposed by Liu et al. (2018), introduces concept convergence and word divergence derived from hypernym-hyponym relations of WordNet to *word2vec*. We set both context window size and the number of negative samples as 10; other parameters remain default.

For *fastText* and *SENSE*, we also try other parameters by changing the number of negative samples or the context window size, but performance is degraded. As reference points, we also present results using only pre-trained embeddings, e.g., *GloVe.6B.300d* from GloVe, *GoogleNews-vectors-negative300* from word2vec, and *wiki-news-300d-1M.vec* from fastText, denoted by *GloVe*$_{pre}$, *word2vec*$_{pre}$, and *fastText*$_{pre}$, respectively. Results of other approaches are not listed here, either because they are similar to the chosen baselines (Liu et al. 2015; Bollegala et al. 2016; Glavaš and Vulić 2018), or they cannot efficiently handle millions of semantic lists (Niu et al. 2017).

The ***default setting*** for WESEK is similar to *word2vec*; both the window size and the number of negative samples are 10, and the number of iterations is 3. For the semantic knowledge step in WESEK, we sample 1 positive neighbor for each target word from the semantic similarity graph and draw 10 negative samples. We set $\lambda = 0.1$, and experimental results show that WESEK has robust performance when λ is less than 0.4, with $\lambda = 0.1$ achieving slightly better performance. We run WESEK and the baselines 10 times under the same settings and report the averaged performance.

5.2 Word Similarity

We use the following datasets to evaluate word similarity: MC30 (Miller and Charles 1991), MEN-TR (Bruni et al. 2012), MTurk771 (Halawi et al. 2012), RG65 (Rubenstein and Goodenough 1965), RW (Luong, Socher, and Manning 2013), WS353-ALL (Finkelstein et al. 2001), WS353-REL, and WS353-SIM (Agirre et al. 2009). These datasets contain human-assigned similarity judgements for all word pairs. We calculate the cosine similarity between word pairs and report the Spearman's rank correlation coefficient $\times 100$ between the rankings produced by cosine similarity and human judgments. The results are shown in Table 2. The comparisons suggest that:

- With the same training data, WESEK always achieves better performance than *word2vec*, which means that leveraging semantic knowledge helps improve word representation learning and WESEK is capable of encoding web semantic knowledge effectively.
- Compared with *fastText* and *GloVe* trained on the Wikipedia dump, WESEK has the best performance on most datasets, except MEN-TR-3k and RW-STANFORD, on which *fastText* has a better performance, mainly for the reason that *fastText* learns sub-word representations that better handle rare words or out-of-vocabulary words.
- Compared with other word representation specialization methods that leverage WordNet or PPDB, WESEK also achieves better performance on most datasets. It should be noted that WordNet or PPDB are high-quality semantic resources while the semantic knowledge incorporated in WESEK is fully automatically constructed from web data. The results explain that our web semantic knowledge has relatively high quality.
- Compared with pre-trained embeddings trained on much more data, WESEK trained on a Wikipedia dump obtains similar performance, which highlights that introducing semantic knowledge helps achieve comparable performance more efficiently.

5.3 Word Sense Disambiguation

Word sense disambiguation (WSD) aims to assign predefined senses to words in contexts. Iacobacci, Pilehvar, and Navigli (2016) provide a study of various techniques to combine word embeddings with standard WSD features to train supervised learning models. They show that word embeddings introduced with exponential decay, which gives more importance to the closer context, achieves the best performance. We follow the same settings as (Iacobacci, Pilehvar, and Navigli 2016), with 400 as embedding size and other parameters are under ***default settings***. We then use exponential decay to feed WESEK to the IMS system proposed by Zhong and Ng (2010). The IMS makes use of context words, POS tags of the context words, and some local collocation features. We perform evaluation on Senseval-2 (Edmonds and Cotton 2001), Senseval-3 (Mihalcea, Chklovski, and Kilgarriff 2004), and SemEval-2007 (Pradhan et al. 2007) English lexical sample WSD tasks, denoted by SE2, SE3, and SE7, respectively.

We list all baselines in (Iacobacci, Pilehvar, and Navigli 2016), i.e., Taghipour and Ng (2015), AutoExtend (Rothe and Schütze 2015), C&W (Collobert and Weston 2008), and retrofitting (Faruqui et al. 2015). In addition, we also add the baselines as mentioned above with exponential decay. To make comparisons fair, for the baselines, the size of embeddings is also set to 400, and other parameters remain unchanged. The results in Table 3 show that WESEK achieves the best performance on all three benchmarks. Such results suggest that WESEK successfully captures word senses implicitly contained in semantic lists. For example, in the set of semantic lists, "apple" might frequently co-occur with "orange" as a fruit, and "microsoft" as a company. When leveraging the semantic graph for WRL, WESEK aims to assure that distances between "apple" and "orange", "apple" and "microsoft" are both close, which helps to improve WSD performance.

	SE2	SE3	SE7
IMS	65.3	72.9	87.9
Taghipour and Ng (2015)	66.2	73.4	-
AutoExtend	66.5	73.6	-
IMS + C&W	64.3	70.1	88.0
IMS + Retrofitting	65.9	72.8	88.3
IMS + word2vec	69.9	75.2	89.4
IMS + CF	66.3	73.8	88.6
IMS + AR	66.8	73.6	88.6
IMS + PostSpec$_{\mathrm{CF}}$	66.3	73.8	88.7
IMS + PostSpec$_{\mathrm{AR}}$	66.4	73.6	88.6
IMS + SENSE	69.8	75.4	89.5
IMS + fastText	69.7	75.4	89.7
IMS + WESEK	**70.1**	**75.5**	**89.8**

Table 3: Lexical word sense disambiguation.

5.4 Text Classification and Textual Similarity

We use SentEval (Conneau and Kiela 2018) to evaluate WESEK on text classification and textual similarity. SentEval is a toolkit to evaluate the quality of sentence representations. In SentEval, sentence embeddings are considered as input features for various downstream NLP tasks, including text classification, natural language inference, and semantic textual similarity, etc. We use the bag-of-words (average of word vectors) (BOW) provided by the SentEval toolkit to obtain sentence embeddings for evaluation. Though we can try other models, we believe that simpler models might better reflect the effectiveness of WESEK.

Text classification. In SentEval, classification tasks include sentiment analysis (MR, SST, CR and MPQA) (Pang and Lee 2005; Socher et al. 2013; Hu and Liu 2004; Wiebe, Wilson, and Cardie 2005), question-type classification (TREC) (Voorhees and Tice 2000), subjectivity/objectivity classification (SUBJ) (Pang and Lee 2004), entailment (SICK-E) (Marelli et al. 2014), and paraphrase classification on Microsoft Research Paraphrase Corpus (MRPC) (Dolan, Quirk, and Brockett 2004). The performance is evaluated on the logistic regression classifier trained on the BOW embeddings. We report the F1 score for MRPC and the accuracy for other tasks.

Semantic textual similarity. For semantic textual similarity (STS) tasks, we make evaluations on semantic relatedness tasks (SICK-R and STS-B) (Cer et al. 2017), and STS tasks from 2012 to 2016 (Agirre et al. 2012; 2013; 2014; 2015; 2016). We calculate the Pearson correlations$\times$100 between the similarity of sentence representations obtained by BOW and human judgement.

Table 4 presents the results for text classification and semantic textual similarity. WESEK achieves the best overall performance for both tasks. The results also show that:

	Retro_{WN}	$\text{Retro}_{\text{PPDB}}$	CF	AR	$\text{PostSpec}_{\text{CF}}$	$\text{PostSpec}_{\text{AR}}$	SENSE	GloVe	word2vec	fastText	WESEK
MR	73.4	73.6	73.2	73.3	72.7	74.2	74.6	72.9	74.8	74.1	**75.3**
CR	72.8	74.2	75.1	75.5	76.7	76.7	76.1	75.2	75.8	75.8	**76.5**
SUBJ	90.0	89.7	89.6	89.0	89.6	89.9	90.9	90.0	90.8	90.8	**91.0**
MPQA	86.7	87.4	86.7	87.6	87.1	87.8	86.6	85.6	86.9	87.1	86.8
SST2	79.6	79.4	78.4	77.4	79.4	80.0	79.0	76.6	78.8	78.6	78.7
SST5	40.7	41.6	41.1	39.0	41.6	41.2	41.5	39.6	41.7	40.9	40.9
TREC	64.3	65.0	67.0	68.0	66.8	71.4	72.6	69.0	71.8	66.0	**72.8**
MRPC	80.1	79.8	78.8	79.2	80.2	79.9	80.1	79.1	79.8	78.8	**80.4**
SICK-E	76.6	76.4	78.1	76.3	78.7	78.8	77.3	77.2	78.3	78.2	**78.4**
[AVG]	73.8	74.1	74.2	73.9	74.8	75.5	75.4	73.9	75.4	74.5	**75.6**
STS12	45.5	53.3	49.7	52.4	52.0	52.0	56.3	40.7	57.4	58.0	**58.7**
STS13	44.3	50.6	42.0	44.2	44.9	42.9	54.0	35.6	53.9	55.1	**56.0**
STS14	52.9	57.0	51.7	52.3	55.9	55.2	62.0	43.2	62.3	63.0	**64.9**
STS15	51.8	55.5	53.0	55.4	54.1	53.8	62.2	44.1	61.8	61.9	**64.6**
STS16	40.2	49.0	47.0	48.8	44.4	47.4	56.3	31.9	57.8	58.6	**61.2**
SICK-R	75.1	76.2	75.3	72.8	77.1	76.8	78.2	76.6	79.4	79.4	79.3
STS-B	57.4	61.2	58.2	52.9	61.8	61.1	64.6	58.6	64.2	**64.7**	**64.7**
[AVG]	52.5	57.5	53.8	54.1	55.7	55.6	61.9	47.2	62.4	63.0	**64.2**

Table 4: Text classification (top half) and textual similarity (bottom half). [AVG] means the average over all tasks. The numbers in **bold** mean that WESEK outperforms *word2vec*, *GloVe*, and *fastText*. The underlined numbers denote the best performance.

- For text classification, WESEK achieves better performance on most datasets compared with each of the baselines. WESEK does not perform as well as *PostSpec*$_{CF}$ and *PostSpec*$_{AR}$ on sentiment classification tasks (CR, SST, and MPQA), mainly because many antonyms introduced in these methods have opinion polarities. For other classification tasks, the improved results demonstrate that the semantic representations are enhanced in WESEK.

- On the STS tasks, WESEK significantly outperforms *word2vec*, *GloVe*, and *fastText*, which explains that semantic knowledge mined from the web can complement and enhance the semantic representations learned from text corpora. Other approaches that leverage semantic lexicons or linguistic constraints have poor performance. The reason might be that word representations lose the original captured semantic regularities in text during specialization, or the introduced semantic lexicons/linguistic constraints have limited coverage of the vocabulary.

6 Conclusion and Future Work

Enhancing WRL with semantic lexicons is a hot topic in NLP research areas. In this paper, we propose WESEK - an approach to enhance word embeddings with semantic knowledge. Unlike existing work that leverages WordNet or PPDB, the semantic knowledge introduced in WESEK can be **fully automatically** constructed from web data. WESEK outperforms state-of-the-art embeddings on multiple NLP tasks, which demonstrates that leveraging web semantic knowledge helps improve WRL and that WESEK is capable of encoding semantic knowledge effectively.

In future work, we will apply WESEK to downstream tasks of other languages. We also intend to further investigate semantic relations like hypernyms and hyponyms, synonyms, antonyms that can be automatically, or partially automatically, constructed from the web to improve WRL.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China (Grand Nos. U1636211, 61672081, 61370126), Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001) and National Key R&D Program of China (No.2016QY04W0802).

References

Agirre, E.; Alfonseca, E.; Hall, K.; Kravalova, J.; Paşca, M.; and Soroa, A. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL-HLT*.

Agirre, E.; Diab, M.; Cer, D.; and Gonzalez-Agirre, A. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *SemEval@NAACL-HLT 2012*.

Agirre, E.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; and Guo, W. 2013. * sem 2013 shared task: Semantic textual similarity. In *SemEval@NAACL-HLT 2013*.

Agirre, E.; Banea, C.; Cardie, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Guo, W.; Mihalcea, R.; Rigau, G.; and Wiebe, J. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *SemEval@COLING 2014*.

Agirre, E.; Banea, C.; Cardie, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Guo, W.; Lopez-Gazpio, I.; Maritxalar, M.; Mihalcea, R.; et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *SemEval@NAACL-HLT 2015*.

Agirre, E.; Banea, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Mihalcea, R.; Rigau, G.; and Wiebe, J. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval@NAACL-HLT 2016*.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *TACL*.

Bollegala, D.; Alsuhaibani, M.; Maehara, T.; and Kawarabayashi, K.-i. 2016. Joint word representation learning using a corpus and a semantic lexicon. In *AAAI*.

Bruni, E.; Boleda, G.; Baroni, M.; and Tran, N.-K. 2012. Distributional semantics in technicolor. In *ACL*.

Cer, D.; Diab, M.; Agirre, E.; Lopez-Gazpio, I.; and Specia, L. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval-2017*.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.

Conneau, A., and Kiela, D. 2018. Senteval: An evaluation toolkit for universal sentence representations. *LREC*.

Dolan, B.; Quirk, C.; and Brockett, C. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING*.

Edmonds, P., and Cotton, S. 2001. Senseval-2: overview. In *SENSEVAL@ACL 2001*.

Faruqui, M.; Dodge, J.; Jauhar, S. K.; Dyer, C.; Hovy, E.; and Smith, N. A. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL-HLT*.

Finkelstein, L.; Gabrilovich, E.; Matias, Y.; Rivlin, E.; Solan, Z.; Wolfman, G.; and Ruppin, E. 2001. Placing search in context: The concept revisited. In *WWW*.

Glavaš, G., and Vulić, I. 2018. Explicit retrofitting of distributional word vectors. In *ACL*.

Halawi, G.; Dror, G.; Gabrilovich, E.; and Koren, Y. 2012. Large-scale learning of word relatedness with constraints. In *SIGKDD*.

Hearst, M. A. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*.

Hu, M., and Liu, B. 2004. Mining and summarizing customer reviews. In *SIGKDD*.

Iacobacci, I.; Pilehvar, M. T.; and Navigli, R. 2016. Embeddings for word sense disambiguation: An evaluation study. In *ACL*.

Kiela, D.; Hill, F.; and Clark, S. 2015. Specializing word embeddings for similarity or relatedness. In *EMNLP*.

Liu, Q.; Jiang, H.; Wei, S.; Ling, Z.-H.; and Hu, Y. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *ACL-IJCNLP*.

Liu, Q.; Huang, H.; Zhang, G.; Gao, Y.; Xuan, J.; and Lu, J. 2018. Semantic structure-based word embedding by incorporating concept convergence and word divergence. In *AAAI*.

Luong, T.; Socher, R.; and Manning, C. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*.

Marelli, M.; Menini, S.; Baroni, M.; Bentivogli, L.; Bernardi, R.; Zamparelli, R.; et al. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*.

Mihalcea, R.; Chklovski, T.; and Kilgarriff, A. 2004. The senseval-3 english lexical sample task. In *SENSEVAL@ACL 2004*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Miller, G. A., and Charles, W. G. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*.

Mrkšić, N.; OSéaghdha, D.; Thomson, B.; Gašic, M.; Rojas-Barahona, L.; Su, P.-H.; Vandyke, D.; Wen, T.-H.; and Young, S. 2016. Counter-fitting word vectors to linguistic constraints. In *NAACL-HLT*.

Mrkšić, N.; Vulić, I.; Séaghdha, D. Ó.; Leviant, I.; Reichart, R.; Gašić, M.; Korhonen, A.; and Young, S. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *TACL*.

Nguyen, K. A.; Köper, M.; im Walde, S. S.; and Vu, N. T. 2017. Hierarchical embeddings for hypernymy detection and directionality. In *EMNLP*.

Niu, Y.; Xie, R.; Liu, Z.; and Sun, M. 2017. Improved word representation learning with sememes. In *ACL*.

Ono, M.; Miwa, M.; and Sasaki, Y. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *NAACL-HLT*.

Osborne, D.; Narayan, S.; and Cohen, S. 2016. Encoding prior knowledge with eigenword embeddings. *TACL*.

Pang, B., and Lee, L. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*.

Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*.

Pasca, M. 2004. Acquisition of categorized named entities for web search. In *CIKM*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Pradhan, S. S.; Loper, E.; Dligach, D.; and Palmer, M. 2007. Semeval-2007 task 17: English lexical sample, srl and all words. In *SemEval@ACL 2007*.

Rothe, S., and Schütze, H. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *ACL-IJCNLP*.

Rubenstein, H., and Goodenough, J. B. 1965. Contextual correlates of synonymy. *CACM*.

Shi, S.; Zhang, H.; Yuan, X.; and Wen, J.-R. 2010. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *COLING*.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Taghipour, K., and Ng, H. T. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *NAACL-HLT*.

Voorhees, E. M., and Tice, D. M. 2000. Building a question answering test collection. In *SIGIR*.

Vulić, I.; Mrkšić, N.; Reichart, R.; Séaghdha, D. Ó.; Young, S.; and Korhonen, A. 2017. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *ACL*.

Vulić, I.; Glavaš, G.; Mrkšić, N.; and Korhonen, A. 2018. Post-specialisation: Retrofitting vectors of words unseen in lexical resources. In *NAACL-HLT*.

Wiebe, J.; Wilson, T.; and Cardie, C. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*.

Wu, W.; Li, H.; Wang, H.; and Zhu, K. Q. 2012. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*.

Yu, M., and Dredze, M. 2014. Improving lexical embeddings with semantic knowledge. In *ACL*.

Zhang, H.; Zhu, M.; Shi, S.; and Wen, J.-R. 2009. Employing topic models for pattern-based semantic class discovery. In *ACL-IJCNLP*.

Zhang, Z.; Zhu, K. Q.; Wang, H.; and Li, H. 2013. Automatic extraction of top-k lists from the web. In *ICDE*.

Zhong, Z., and Ng, H. T. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL*.