Task Independent Fine Tuning for Word Embeddings

Xuefeng Yang, *Member, IEEE*, and Kezhi Mao, *Member, IEEE*

***Abstract*—Representation learning of words, also known as word embedding technique, is based on the distributional hypothesis that words with similar semantic meanings have similar context. The selection of context window naturally has an influence on word vectors learned. However, it is found that the word vectors are often very sensitive to the defined context window, and unfortunately there is no unified optimal context window for all words. One impact of this issues is that, under a predefined context window, the semantic meanings of some words may not be well represented by the learned vectors. To alleviate the problem and improve word embeddings, we propose a task-independent fine-tuning framework in this paper. The main idea of the task-independent fine tuning is to integrate multiple word embeddings and lexical semantic resources to fine tune a target word embedding. The effectiveness of the proposed framework is tested by tasks of semantic similarity prediction, analogical reasoning, and sentence completion. Experiments results on six word embeddings and eight datasets show that the proposed fine-tuning framework could significantly improve word embeddings.**

***Index Terms*—Fine tuning, word embedding, word representation learning.**

I. Introduction

REPRESENTATION leaning of words, also known as word embedding technique, aims to learn numerical vectors to represent semantic meanings of words. Vector representation of words could facilitate a variety of tasks in computational linguistics and natural language processing, including machine translation [1], [2], language modeling [3], [4], language understanding [5] and dependency parsing [6] etc. Word vectors are usually learned by two types of models: global context-based models and local context-based models. In the global context-based models, the statistics of contextual information is first summarized in a high dimensional sparse co-occurrence matrix, and a dimensionality reduction technique such as singular value decomposition (SVD) is then applied to the co-occurrence matrix to produce low dimensional representations of words [7], [8]. By contrast, the local context-based models learn word vectors using neural language models, i.e. artificial neural networks. The neural language model was firstly proposed in [9], and since then many variants have been developed [10]–[14].

Manuscript received March 7, 2016; revised June 9, 2016 and December 17, 2016; accepted December 19, 2016. Date of publication December 23, 2016; date of current version March 14, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Min Zhang.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: yang0302@e.ntu.edu.sg; ekzmao@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2016.2644863

TABLE I
TOP 5 MOST SIMILAR WORDS TO "SNAKE".

Window size 5	Window size 7	Window size 9
tarantula	cobra	cobra
venomous	tarantula	lizard
frog	nonvenomous	tarantula
rattlesnake	spider	frog
lizard	python	porcupine

In both global and local context-based models, the word vectors learned are influenced by the defined context window. We found that a slight change in the context window size could cause unexpected changes in the word vectors. To illustrate this problem, we use word "snake" as an example. Table I lists the top five words whose vector representations have the highest cosine similarity scores to the vector representation of "snake" under different context window size in the representation learning.

Word representation learning is based on the distributional hypothesis that words with similar meanings have similar context [15], [16]. The defined context window size naturally has an influence on the word vectors learned. But the example shown in Table I reveals that the word vectors are very sensitive to the context window. Since there is no unified optimal window size for all words, an immediate effect of the issue is that, in a word embedding, some words are well represented while other are not. In this paper, we aim to alleviate this problem and improve word embeddings by using task independent pseudo supervised fine tuning.

Fine tuning, as a post-processing, has emerged as a popular technique to improve word embeddings. Fine tuning usually employs task specific training data, and the fine-tuned word embedding is limited to a particular task.

To improve word embeddings for general purpose, it is necessary to develop a task-independent fine tuning framework. Different from task-dependent fine tuning, the task-independent fine tuning should be based on task-independent knowledge. Since most existing word embeddings are learned from various task-independent corpora, these word embeddings are good sources of task independent knowledge. Motivated by this consideration, an intuitive solution to the task-independent fine tuning problem is to fine tune a target word embedding using other word embeddings. However, the use of other word embeddings as the task independent knowledge source is not straightforward. What knowledge of other word embeddings is usable and how to extract this knowledge are all questions to be answered.

To use supervised learning paradigm to fine tune word vectors, there are two additional issues. One issue is the generation

of labeled task independent training data. Manually labelling a large number of training data is impractical. In addition, the exact measure of semantic similarity is hard to determine since a word pair may have very different cosine similarity scores under different word embeddings. For the same word pair, even linguists may give different similarity judgements. Another issue is the design of a suitable learning algorithm to avoid any major disruptions to the original word embedding by the fine tuning.

The task independent fine tuning framework proposed in this paper provides solutions to the above mentioned issues. First, an algorithm that automatically generates labeled training data by integrating knowledge from multiple source word embeddings and lexical semantic resources is proposed. The training data of a training word is a set of semantically similar or related words to the training word, and the labels of the training data are defined as the rankings, i.e. positions of the words in the set sorted by their similarity scores to the training word. Second, an inverse error weighted mini-batch stochastic gradient descent optimization algorithm is developed to effectively update the target word embedding and meanwhile avoid any major disruptions to the target word embedding under fine tuning.

The rest of the paper is organized as follows. In Section II, related work on word representation learning and fine tuning is briefly reviewed. The proposed task independent fine tuning framework is detailed in Section III. Experimental evaluation and analysis are given in Section IV. Section V concludes the paper.

II. Related Work

Vector representation learning of words has received considerable attentions in recent years. The methods reported in the literature can be classified into two categories: the global context-based models and the local context-based models.

In the global context-based models, the statistical contextual information is usually summarized into a co-occurrence matrix. Each row of the co-occurrence matrix specifies a word's co-occurring words and their normalized occurrence frequencies in the training corpus, representing the global contextual information of the word. The co-occurrence matrix is usually high dimensional and sparse, and a dimensionality reduction technique such as singular value decomposition (SVD) is applied to the high dimensional sparse matrix to generate a low dimensional dense matrix with the same number of rows as the original matrix [7], [8]. Each row of the low dimensional matrix is the vector representation of a word.

The local context-based models are usually based on artificial neural networks, whose input is the context of a word and whose output is the vector representation of the word. The artificial neural network is trained in terms of the adopted training target and optimization algorithm. A variety of training targets have been proposed. For example, [9] employs a sequence of words to predict the next word, while [10] employs context to predict the middle word, and [12] employs the contral word to predict all context words. The commonly used optimization algorithms include hierarchical softmax [17], hierarchical tree construction [13], [18], and negative sampling [11], [13], [19] etc.

Word embeddings are usually trained on large unlabeled training corpora. When applied to solve a specific problem, word embeddings are often fine tuned on the task-specific training data. For example, a lexicon infused word representation learning algorithm built upon the existing skip-gram model is proposed in [20] for named entity recognition. [21] trains sentiment specific word embedding on the labeled sentiment data, and [22] tailors existing word embeddings for dependency parsing. All these works adopt task-dependent fine tuning to adapt existing embeddings to specific tasks. [23] explores the contrasting meaning word embedding so that the learned embedding may encode the contrasting meaning information available in lexical semantic resources. The contrasting meaning embedding is a case that may not be applied in our proposed fine tuning framework because nearly all other embeddings employed in our framework mainly encode semantic similarity and relatedness information. The task independent fine tuning can be combined with task dependent fine tuning to provide a better initialization to the latter.

However, task specific training data is not available in some applications such as semantic similarity and relatedness problems for words, phrases, sentences and documents. In such scenarios, we have to resort to task-independent fine tuning, which incorporates task independent knowledge extracted from lexical semantic resources into the fine tuning process. For example, [24] proposes a post-processing algorithm to encode the information from knowledge resource into the word embedding. [23], [25] employ knowledge in lexical semantic resources to tackle the contrasting meaning problem. However, these approaches only use the knowledge in lexical semantic resources, while the fine tuning framework proposed in our paper employs both lexical semantic resources and source word embeddings.

Lexical knowledge resources can also be used directly as auxiliary signal in the training process. For example, [26] proposes a framework to add relational and categorical knowledge as regularization of the original training target, and [27] utilizes the morphological knowledge as both additional input representation and auxiliary supervision. [28] trains a compositional morphological vector representation, in which a word vector is the sum of its morphological vectors.

In [29], a stochastic neighbor embedding is developed for dimensionality reduction of images and document count matrices, with the goal of transferring the relationship of neighbours in the original high dimensional space to the low dimensional space. By contrast, our work aims to transfer the ranking information of source word embeddings and lexical semantic resources to the target word embedding under fine tuning.

III. Task Independent Fine Tuning for Word Representations

A. Overview

To improve word embeddings, this study proposes a task independent fine tuning framework as shown in Fig. 1.

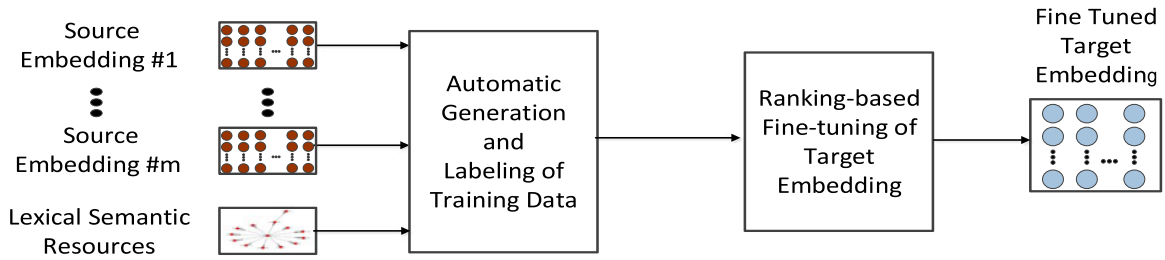


Fig. 1. Task independent fine tuning framework.

The proposed task independent fine tuning framework is built upon the existing word embeddings and lexical semantic resources. The fine tuning framework has two main modules, including automatic generation and labeling of training data, and ranking-based word embedding fine tuning.

In the module of training data generation and labeling, knowledge of multiple source word embeddings and lexical semantic resources is integrated to generate and label the training data for each training word whose vector representation is to be fine tuned. The source embeddings are some well known embeddings such as SENNA50 [11], GloVe300 [14], and Word2vec [13] etc. The lexical semantic resources can be WordNet [30] etc. The training data of a training word refers to a set of semantically most similar or related words to the training word. The target values or labels of the training data are defined as the rankings, i.e. the positions of the words in the set sorted by their similarity measures to the training word. With the labeled training data, the fine tuning module adjusts the word vectors in the target word embedding, with the objective of minimizing the discrepancy between the rankings obtained from the target word embedding and the rankings extracted from the multiple source word embeddings and lexical semantic resources. The target word embedding refers to the embedding under fine tuning. The source embeddings may also be a target embedding since the current word embeddings are far from perfect and may be further improved by the proposed task independent fine tuning framework.

B. Automatic Generation and Labeling of Training Data

As aforementioned, the pseudo supervised fine tuning framework adjusts word vectors of the target word embedding using labeled training data. How to automatically generate the training data, and how to automatically label the training data are two issues to be addressed next.

1) Why Ranking?: In task independent fine tuning, semantic similarity may be used as the training target. However, we found that the semantic similarity measure between two words could be affected by a few factors such as the dimensionality of the word embedding. Take GloVe word embedding [14] as an example. The cosine similarity measure between words "fish" and "salmon" in the 300-dimensional embedding is 0.6596, while the measure is 0.8340 in the 50-dimensional embedding. Although the similarity scores are quite different, the word "salmon" is the most similar word to word "fish" in both embeddings. This example reveals that similarity ranking, as an indicator of the similarity between two words, may be more robust than similarity score. Inspired by this finding, we propose to employ ranking as the target value or label of the training data in the proposed task independent fine tuning framework.

TABLE II
ERRORS IN GLOVE300 EMBEDDING

Hill	Run	Paper
now, **57**	**three**, **12**	**instead**, **30**
known, **64**	**only**, **17**	**put**, **33**
woods, 70	**not**, **30**	document,35
hillside, 72	come, 39	books, 42
called, **74**	**seven**, **47**	**mirror**, **53**
wood, 97	take, 58	publish, 57

2) Ranking Information Extraction from Multiple Knowledge Sources: The similarity ranking can be extracted from any of the task independent knowledge sources, including the source word embeddings and the lexical semantic resources. But the question is which knowledge source to use. Even the state-of-art embeddings may not always provide reliable ranking.

Table II shows some identified similar words to "hill", "run", and "paper" respectively, based on GloVe300 word embedding. The numerical value beside a word denotes the ranking, i.e. the position in the list of semantically similar or related words to the three words. Obviously, even for the three frequently used words, the rankings could be inaccurate. For example, for word "hill", words "now" and "known" are incorrectly ranked higher than word "wood". To alleviate this problem, we propose to rank the similarity of words to a training word based on lexical semantic resources and multiple source word embeddings trained with different algorithms, different context window and corpora. It is believed that fusion of knowledge from multiple sources could provide more reliable ranking information.

For a training word, the multiple knowledge source-based ranking information extraction involves 3 steps. A metrics is first used to measure the similarity score of every word in the vocabulary to the training word using each of the source word embeddings and lexical semantic resources. A fusion criterion is then employed to integrate the multiple scores into one consensus score. A ranking procedure is finally performed to sort the words based on their consensus semantic similarity scores to the training word.

Assuming ω_t denotes a training word, ω_v denotes an arbitrary word in the vocabulary, and $\mathcal{K}_i$ denotes the i^{th} knowledge source

(a source word embedding or a lexical semantic resource). The normalized similarity measure of word ω_v to training word ω_t based on knowledge source $\mathcal{K}_i$ is given by:

$$s_i(\omega_t, \omega_v) = \frac{s'_i(\omega_t, \omega_v)}{\max\{s'_i(\omega_t, \omega_v) | \omega_v \in V_i, \omega_v \neq \omega_t\}} \quad (1)$$

where V_i is the vocabulary of knowledge source $\mathcal{K}_i$. If $\mathcal{K}_i$ is a source word embedding, $s'_i(\omega_t, \omega_v)$ is the commonly used cosine similarity measure. If $\mathcal{K}_i$ is the lexical semantic resource WordNet, $s'_i(\omega_t, \omega_v)$ is Lin's measure [31]. In the proposed fine tuning framework, multiple source word embeddings and lexical semantic resources are employed, and the normalized similarity of word ω_v to word ω_t can be obtained from each of the employed knowledge sources. It is noted that the source word embeddings and lexical semantic resources have different-sized vocabulary, and the out-of-vocabulary problem is a potential issue, i.e. word ω_t or ω_v is not in the vocabulary of a knowledge source. If the out-of-vocabulary problem occurs in knowledge source $\mathcal{K}_i$, we set $s'_i(\omega_t, \omega_v) = 0$. The consensus score can be obtained by the fusion of the multiple scores obtained from multiple knowledge sources:

$$S(\omega_t, \omega_v) = \frac{\sum_i s_i(\omega_t, \omega_v)}{\sum_i I((\omega_t, \omega_v) \in V_i)} \quad (2)$$

where I is the indicator function whose value is 1 if the input condition is satisfied, and is 0 otherwise. This indicator actually says that if ω_t or ω_v is out of the vocabulary of $\mathcal{K}_i$, the fusion of similarity scores of ω_v to ω_t will not take the result of knowledge source $\mathcal{K}_i$ into account.

The consensus score represents the overall semantic similarity or relatedness of word ω_v to training word ω_t. For each word in the vocabulary, such a consensus score can be obtained. The words are then sorted according to the consensus scores. The final outcome is the list of semantic similar words to training word ω_t, and the rankings or positions of the words in the list. This outcome is referred to as ranking information of training word ω_t in this paper.

3) Selection and Labeling of Training Data: A word is semantically similar or related to only a small portion of the entire vocabulary. It does not make much sense to rank the similarity of two words "pen" and "water" to "basketball" since both words are not similar or related to "basketball". Based on this insight, we believe that the fine tuning process just needs to take care of the meaningful word pairs with strong relations. In other words, for each training word ω_t, only a set of the most similar or related words, denoted by $D_{\omega_t} = \{\omega_{v_i}, i = 1, 2, \ldots, N\}$, are needed to fine tune ω_t. The set of the most similar words D_{ω_t} can be easily obtained by selecting the top ranked words based on the ranking information previously obtained. This set of the most similar or related words D_{ω_t} is defined as the training data of ω_t, and the rankings of these words in the set are defined as the target values or labels of the training data. It is found that several hundred most similar or related words, say 200, are enough for each training word. By using the short list of semantically most similar or related words, the computational cost could be significantly reduced.

Algorithm :1 Training Data Generation and Labeling.

Input: VocabList V stores words index in a list
Count Matrix D corresponds to the words in V
Matrix C is the times of word pair selected
Matrix A is the times of word pair appearing
$\mathcal{K}$ is the list of knowledge sources
N is the number of training data for each training word
$data$ is a mapping structure that stores the output
Initialize $D = 0, C = 0$
1 Calculate the score of data
for all $\mathcal{K}_i$ in $\mathcal{K}$ **do**
  for all w_t in V **do**
    $(\mathbf{w}_N, \mathbf{s}_N) \leftarrow$ top N of $\{s_i(w_t, w_v) | w_v \in V\}$
    $s_{max} \leftarrow max(\mathbf{s}_N)$
    for all w_v and s_i in $\mathbf{w}_N, \mathbf{s}_N$ **do**
      $D[w_t, w_v] += s_i(w_t, w_v)/s_{max}$
      $C[w_t, w_v] += 1$
    end for
  end for
end for
2 Remove low score data
for all w_t in V **do**
  for all w_v in V **do**
    if $C[w_t, w_v] <= 2$ **then**
      $D[w_t, w_v] = 0$
    end if
  end for
end for
3 Remove bias of frequency
$D = D/A$ hint: point wise division
for all w_t in V **do**
  $data[w_t] \leftarrow$ select nonzeros in $D[w_t]$ and sort
end for
Output: $data$ stores the ranking of related words

4) Algorithm: The procedure of automatic generation and labeling of training data is given in Algorithm 1 (Our implementation is based on sparse matrix and nested mapping because the size of matrix is very large):

C. Ranking-Based Fine Tuning

1) Training Target: Maximizing the context prediction capability is often used as the training target of neural language models for word vector learning. In the proposed task independent fine tuning framework, the training target is to minimize the discrepancy between the rankings extracted from the target embedding and rankings extracted from the source embeddings and lexical semantic resources.

To achieve the above goal, a ranking loss function could be employed as the minimization cost function:

$$J_{rank}(\omega_t, \omega_v) = \sum_{\omega_t \in T} \sum_{\omega_{v_i} \in D_{\omega_t}} | L_{\omega_{v_i}} - R_{\omega_{v_i}} |^2 \quad (3)$$

where T is the set of training words, $D_{\omega_t} = \{\omega_{v_i}, i = 1, 2, \ldots, N\}$ is the training data of word ω_t, $L_{\omega_{v_i}}$ denotes the ranking of ω_{v_i} extracted from multiple knowledge sources at the

stage of training data generation and labeling, and $R_{\omega_{v_i}}$ denotes the ranking extracted from the target word embedding under fine tuning.

Because the ranking loss function is non-differentiable, we choose to minimize the difference of semantic similarity scores at the two ranking positions:

$$J(\omega_t, \omega_v) = \sum_{\omega_t \in T} \sum_{\omega_{vi} \in D_{\omega_t}} | S(L_{\omega_{v_i}}) - S(R_{\omega_{v_i}}) |^2 \quad (4)$$

Minimizing the difference of similarity measures between the two ranking positions also reduces the ranking loss.

2) Inverse Error Weighted Mini-batch SGD Algorithm: The mini-batch stochastic gradient descent (SGD) algorithm is commonly used in word embedding learning:

$$\omega_t = \omega_t - \eta \times \frac{1}{N} \sum_{i=1}^{N} \frac{\partial J(\omega_t, \omega_{v_i})}{\partial \omega_t} \quad (5)$$

where N is the mini-batch size, η is the learning rate, and $\frac{\partial J(\omega_t, \omega_{vi})}{\partial \omega_t}$ is the gradient of the loss function when ω_{v_i} is used to adjust ω_t. In this paper, the mini-batch is the training data of a training word, i.e. the short list of the most similar or related words to the training word. It is found the standard mini-batch SGD reduces the training loss, but degrades the performance on all tasks. This appears similar to the overfitting problem in which the model fits the training data perfectly well but performs very bad on the testing data, but the problem here is different. The overfitting problem normally occurs at the final stage of the training and the model has more parameters than necessary, while the problem here happens at the beginning of the training, even the number of parameters is very small.

After exploring the learning outcomes, we found that the root cause of the problem is that the target word embedding is changed too much by the standard mini-batch SGD updating rule. The current word embeddings are still not perfect, and some strongly related words may not be ranked high in the training data due to factors such as polysemy, corpus bias and imbalanced word frequency etc. Such words may have very large ranking error and large gradient since the magnitude of the gradient is proportional to the error, and this in turn results in major changes to the target word embedding.

In this fine tuning framework, the principle that data with larger error deserves more adjustment is not applicable. An inverse error weighted mini-batch SGD algorithm is proposed in this paper to address the problem:

$$\omega_t = \omega_t - \eta \times \frac{1}{N} \sum_{i=1}^{N} \frac{1}{e(\omega_{v_i})} \frac{\partial J(\omega_t, \omega_{v_i})}{\partial \omega_t} \quad (6)$$

where $e(\omega_{v_i})$ denotes the ranking error of ω_{v_i}.

D. Stopping Criterion

To avoid over-tuning, early stopping can be used:

$$\frac{J_{rank}^{(i)} - J_{rank}^{(i+1)}}{J_{rank}^{(i)}} <= \epsilon \quad (7)$$

Algorithm 2: Ranking-Based Fine Tuning Algorithm.

Input: vocabList V stores words index in a list
$data$ is a nested mapping that stores all training data
$data_{w_t}$ and $rank_{w_t}$ are mappings between words and their ranking for w_t in the labeled training data and the embedding under fine tuning respectively
ul is a list that stores all the local update for word w_t
$update$ is the global update vector for word w_t
δ is random threshold and η is learning rate
Initialize $error = len(V) \times d$, $stop = False$
while $stop$!= True **do**
  for all w_t in V **do**
  $data_{w_t} \leftarrow data[w_t]$
  $rank_{w_t} \leftarrow$ get ranking of all words for w_t
  $ul \leftarrow \emptyset$
  for all w_v in $data_{w_t}$ **do**
    $sign \leftarrow I(rank_{w_t}[w_v] - data_{w_t}[w_v])$
    ul.add($sign \times \cos(w_t, w_v) \times w_v$)
  end for
  $ne_{w_t} \leftarrow$ select $rank_{w_t}$ if cosine>δ and $not\ in\ data_{w_t}$
  for all W_v in ne_{w_t} **do**
    ul.add(cos(w_t, w_v) $\times$ w_v)
  end for
 $update \leftarrow$ mean of vectors in ul
  $update \leftarrow$ normalization($update$)
  $w_t = w_t$ - $\eta \times update$
  $w_t =$ normalization(w_t)
 end for
end while

where $J_{rank}^{(i)}$ denotes the overall ranking loss in the i^{th} epoch. If the relative improvement is smaller than a predefined threshold ϵ, the fine tuning will be stopped. In this paper, the stopping criterion is modified to take the performance of the original word embedding into consideration:

$$\frac{J_{rank}^{(i)} - J_{rank}^{(i+1)}}{J_{rank}^{(0)}} <= \epsilon \quad (8)$$

$J_{rank}^{(0)}$ is actually the performance of the target word embedding before fine tuning, i.e. the original word embedding. The threshold ϵ is set according to the dimensionality of word embeddings and the initial performance. This is because the performance of different embeddings is quite different, and the poorly performing word embeddings may have more space to improve while the word embeddings with good initial performance may not be improved much by the fine tuning.

E. Relation With Skip-Gram and Negative Sampling

The training target function maximized in skip-gram model is the softmax given below:

$$p(v|w) = \frac{\exp(v^T w)}{\sum_i \exp({v_i}^T w)} \quad (9)$$

where $p(v|w)$ is the probability to be maximized given the input vector w and output vector v. v_i denotes the output vector of i^{th} word in the vocabulary.

The inner product of two vectors $v^T w$ may be considered as the cosine similarity without normalization. The numerator corresponds to the similarity of the input and output vectors, and the denominator corresponds to the sum of all the similarity values between the input vector and the output vectors of the entire vocabulary.

The vocabulary of a training corpus is usually at the level of several billions, it is therefore impractical to compute the exact denominator for each training word. Negative sampling is often adopted to address this problem by approximating the denominator. Negative sampling reduces the computational complexity by two operations. The first operation is to make the input and output closer, and the second operation is to force the input vector to move away from random samples. The first operation guarantees a larger numerator, while the second operation reduces the denominator. With increased numerator and decreased denominator, the total training target function is increased.

By contrast, we reduce the computational overhead by the way of using a short list of semantically most similar or related words as the training data. For each training word, several hundred training data is enough for the fine tuning of a word vector, and the total training data size may be just a few millions. In addition, one batch per training word is employed to further reduce computations of the denominator. To be specific, in each iteration, the update of a training word is the weighted mean of gradient obtained from all the shortlisted semantically similar or related words.

In skip-gram model, a sampling process is employed because the algorithm itself is unable to decide which pair of words should be moved apart from each other, though a word should be away from most words. The situation is quite different in the ranking fine tuning algorithm, which identifies the negative data from the comparison of two ranking list. Therefore the negative data is discovered by algorithms based on the training data rather than randomly selected.

In summary, the ranking learning may be defined as the fine tuning process of the original negative sampling approximation algorithm. The pseudo supervised paradigm helps to focus on meaningful and important data, and this makes it possible to compute similarity ranking list for a specific word. With the information in labeled training data, the negative data may also be identified without sampling.

IV. Experiments

To evaluate the proposed task independent fine tuning framework, two experiments were conducted.

In the first experiment, six word emebddings including SENNA, Word2vec and GloVe were fine tuned by the proposed task independent fine tuning framework. The six fine tuned word embeddings were then evaluated with 3 tasks including semantic similarity prediction, analogical reasoning and sentence completion.

The effectiveness of the proposed inverse error weighted mini-batch SGD algorithm was demonstrated in the second experiment. Two groups of methods were compared. The first group employs the commonly used standard mini-batch SGD algorithm, and the second group employs the proposed inverse error weighted mini-batch SGD algorithm. Except the updating rules, all other settings were the same for the two comparison groups.

A. Evaluation Tasks and Datasets

1) Semantic Similarity Prediction: The consistency between machine predicted similarity and human annotated similarity is the most commonly used metrics in the evaluation of word embeddings.

In the tasks of semantic similarity prediction, five public available datasets with different characteristics were used. Wordsim353 [32] and RG65 [33] are two most widely used datasets in evaluation of semantic similarity. MEN3000 (denoted by M3K) [34] and Mturk771 [35] are two recently generated large datasets. YP130 [36] is a dataset designed to evaluate semantic similarity between verbs. As in other works, Spearman's correlation was used to measure the consistency between human annotations and machine predictions.

2) Analogical Reasoning: The analogical reasoning task is to test the ability of relational similarity prediction. The Google analogical reasoning dataset [12], denoted as Google_AR, was used in the experiment. The question is in the following format:"Man : Women = King : ?". The answer should be the exact word "Queen". The Microsoft analogical reasoning dataset [37], denoted as MS_AR, was also used in the experiment.

In analogical reasoning, the nearest neighbour of the vector (Women + King - Man) excluding the three words is selected as the candidate answer. If the candidate answer is the same as the true answer, the question is correctly solved. The overall accuracy was adopted as the evaluation metrics for the analogical reasoning task. Since the vocabulary size of the word embedding is not as large as that used in the original study [12], the questions with unknown word were not included.

3) Sentence Completion: The sentence completion challenge data [38], denoted as MS_SC, was used in the experiment for sentence completion task. The sentence completion task is to select a word that is meaningful and coherent in the context of a complete sentence. In each sentence, an infrequent word is chosen as the focus of the question and four candidates are chosen from a list of words. Only the original word is considered as the correct answer to the question.

In this study, the candidate answer was selected based on the average similarity scores between the candidates and all words in the sentence. The word with largest average similarity score was selected as the answer. The overall accuracy was adopted as the evaluation metrics for the sentence completion task.

B. Experiment Settings

In total, six word embeddings were studied in our experiments, including HLBL50 [39], SENNA50 [11], RNNLM640 [40], GloVe300 [14], DocAndDep2000 [41], and Word2vec

TABLE III
STOPPING CRITERION SETTING (ϵ VALUE)

Word embedding	Senna50	Skip50	HLBL50
ϵ value	0.04	0.05	0.004
Word embedding	Glove300	RNNLM640	Dep1000
ϵ value	0.07	0.07	0.10

[13]. All word embeddings are available in the websites[1] except the Word2vec embedding.

A skip-gram model was trained by the Word2vec toolkit using the billion-word language modeling benchmark data. The window size was set to seven and the mini count of words was set to 10. DocAndDep models were constructed from the document model and the dependency model, and were used separately in this study. The document models were not used due to its unsatisfactory initial performance.

The human summarized knowledge resource WordNet was employed in the experiments. The directly connected words and the words in the same synsets were extracted. In the integration process, all relations from WordNet were treated equally by setting their weights to 1.

The learning rate η is set to 0.1 in all experiments for fast convergence because the computation of cosine similarity matrix in each epoch is very expensive. The stopping threshold ϵ was set in terms of dimensionality and performance of the target word embedding. The settings of ϵ for the six word embeddings are given in Table III

C. Results and Analysis

1) Performance Evaluation: The performance of the six word embeddings is given in Table IV. For each word embedding, two results are given: one is the baseline result of the original word embedding before fine tuning, and another is the result of the fine tuned word embedding by the proposed task independent fine tuning framework.

Among the eight datasets in Table IV, the first five datasets belong to the task of semantic similarity prediction, and the numerical values are Spearman's correlation between human annotations and machine predictions. The remaining three datasets belong to tasks of analogical reasoning or sentence completion, and the numerical values are the overall accuracy.

The results in Table IV show that the performance of all word embeddings is significantly improved after the task-independent fine tuning, which proves the effectiveness of the proposed task independent fine tuning framework. It is observed that the performance improvements in the similarity prediction task (the first five datasets) are more significant. This is because the target of fine tuning is the ranking information of training data, which is strongly related to the similarity prediction task. In analogical reasoning and sentence completion tasks, the improvements are still remarkable, though they are not as significant as in the semantic similarity prediction task.

The results in Table IV show that word embedding Glove300 performs best among all the six word embeddings before fine tuning. Glove300 still gains significant improvements in all datasets after fine tuning. This means that strong word embeddings can still effectively transfer knowledge from weak word embeddings and lexical semantic resources in the proposed fine tuning framework.

As shown in Table IV, the performance of the original HLBL50 word embedding is unsatisfactory, but the fine tuned HLBL50 word embedding has similar performance to the fine tuned SENNA50 and Skip50 word embeddings. This again clearly shows that the knowledge of the source word embeddings and lexical semantic resources has been successfully transferred to the target embedding. It is believed that the knowledge transferring capability of the proposed fine tuning framework is a pillar factor of its success.

In the experiment, six word embeddings and lexical semantic resource WordNet were employed as task independent knowledge sources for training data generation and labeling. These word embeddings and lexical semantic resource have different sized vocabulary. In the experiment, the minimum common vocabulary of the six word embeddings and WordNet was used. Thus, some words in the eight datasets are not covered in the vocabulary, i.e. out of the vocabulary. Table V shows the coverage of the eight datasets. The coverage in the analogical reasoning and sentence completion datasets is low, but the coverage in other five datasets is high. The low coverage in the three datasets is due to the out-of-vocabulary issue instead of cherry picking, and the improvements achieved by the fine tuning are not by chance. The out-of-vocabulary problem deserves more attentions, and will be further explored in future research.

2) Comparison of Standard SGD and Inverse Error Weighted SGD: In the second experiment, the standard mini-batch SGD and the inverse error weighted mini-batch SGD were compared. The learning curves of the two algorithms for the six word embeddings on three similarity prediction datasets are shown in Figs. 2, 3 and 4 respectively, where the solid lines correspond to the standard mini-batch SGD algorithm, while the dashed lines correspond to the proposed inverse error weighted mini-batch SGD algorithm. From Figs. 2, 3 and 4, it is observed that the performance improves with the progress of fine tuning if the proposed inverse error weighted mini-batch SGD is adopted. By contrast, if the standard mini-batch SGD is adopted, the fine tuning degrades the performance. This is because the standard mini-batch SGD disrupts the original word embedding, while the proposed inverse error weighted mini-batch SGD algorithm only fine tunes the word embeddings.

Figs. 5 and 6 show the learning curves of six word embeddings for Google and Microsoft analogical reasoning datasets, respectively. The dashed lines correspond to the proposed inverse error weighted mini-batch SGD algorithm, and the solid lines correspond to the standard mini-batch SGD algorithm. For the two datasets of analogical reasoning task, the same trend is observed: the error weighted mini-batch SGD-based fine tuning

[1]Senna:http://ml.nec-labs.com/senna/
RNNLM:http://www.fit.vutbr.cz/imikolov/rnnlm/
HLBL:http://metaoptimize.com/projects/wordreprs/
Glove:http://nlp.stanford.edu/projects/glove/
DocAndDep:http://www.cs.cmu.edu/ afyshe/papers/conll2013/

TABLE IV
PERFORMANCE OF 6 WORD EMBEDDINGS ON 8 DATASETS

Embedding	Senna50		Skip50		HLBL50		Glove300		RNNLM640		Dep1000	
Dataset	before	after	before	after	before	after	before	after	before	after	before	after
WordSim353	0.40	**0.56**	0.53	**0.55**	0.23	**0.53**	0.55	**0.62**	0.30	**0.55**	0.44	**0.59**
Mturk771	0.48	**0.59**	0.52	**0.60**	0.280	**0.59**	0.63	**0.69**	0.40	**0.65**	0.61	**0.70**
RG65	0.49	**0.61**	0.56	**0.60**	0.36	**0.66**	0.74	**0.77**	0.51	**0.65**	0.68	**0.76**
YP130	0.16	**0.42**	0.35	**0.45**	0.23	**0.49**	0.55	**0.59**	0.40	**0.57**	0.56	**0.65**
M3k	0.59	**0.68**	0.65	**0.73**	0.28	**0.62**	0.73	**0.77**	0.46	**0.65**	0.63	**0.73**
Google_AR	0.128	**0.295**	0.202	**0.360**	0.098	**0.275**	0.479	**0.508**	0.324	**0.382**	0.206	**0.255**
MS_AR	0.185	**0.411**	0.286	**0.451**	0.2	**0.441**	0.665	**0.685**	0.507	**0.538**	0.393	**0.429**
MS_SC	**0.329**	0.289	**0.30**	**0.30**	0.26	0.254	0.335	**0.375**	0.259	**0.291**	0.359	**0.376**

TABLE V
COVERAGE OF EMPLOYED DATASETS

Dataset	WordSim353	Mturk771	RG65	YP130	M3K	Google_AR	MS_AR	MS_SC
No. of All Words	353	771	65	130	3000	19544	8000	1040
No. of Words Used	252	736	65	130	2663	3236	448	172

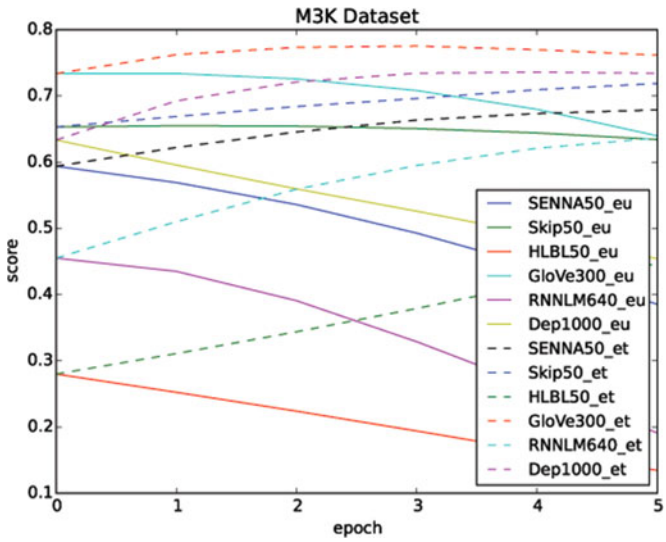


Fig. 2. Comparison of standard SGD and inverse error weighted SGD on M3K similarity prediction dataset.

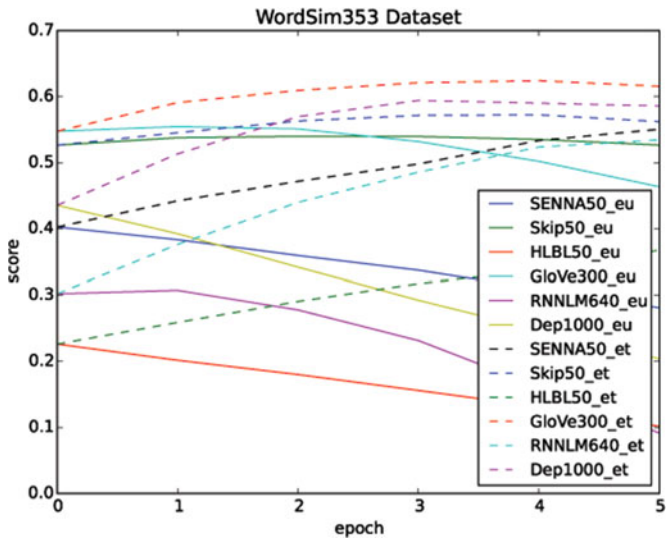


Fig. 3. Comparison of standard SGD and inverse error weighted SGD on WordSim353 similarity prediction dataset.

improves the performance of word embeddings, while the standard SGD-based fine tuning degrades the performance of the original word embeddings.

The results on both the semantic similarity prediction task and the analogical reasoning task show that the proposed inverse error weighted SGD algorithm is another pillar factor to the success of the task independent fine tuning framework.

3) Error Analysis: To better understand the proposed fine tuning framework, an error analysis was conducted on the similarity prediction dataset WordSim353 with Glove300 embedding. Error analysis aims to find the root causes of errors.

The error could be resulted from at least three causes. First, the error could be caused by inaccurate labeling of training data. In the proposed fine tuning framework, the training data of a training word is a list of most similar or related words, and the labels of the training data are the rankings or positions of the words sorted by their similarity scores to the training word. Since the labeling of the training data is done automatically, the labels of some training data are incorrect, which in turn results in error in the fine tuned word embedding.

Second, the error could be caused by the conflicting training target values. Take the word pair "dollar" and "profit" as an example. In the list of words related to "dollar", the word "profit" is ranked 61th, while in the list of words related to "profit", the word "dollar" is ranked 105th.

Third, the error could also be caused by the word wise rather than vocabulary wise ranking learning. The proposed framework fine tunes the word embedding based on the ranking information

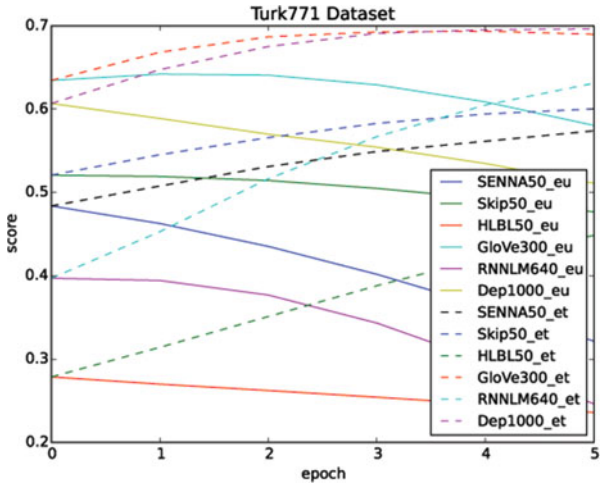


Fig. 4. Comparison of standard SGD and inverse error weighted SGD on Turk771 similarity prediction dataset.

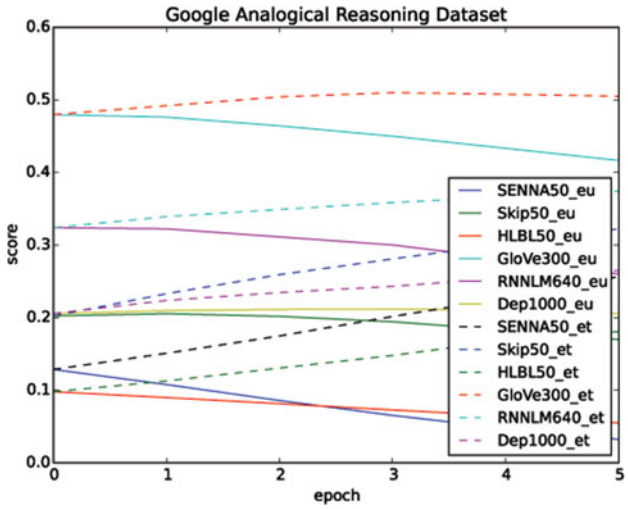


Fig. 5. Comparison of standard SGD and inverse error weighted SGD on Google analogical reasoning dataset.

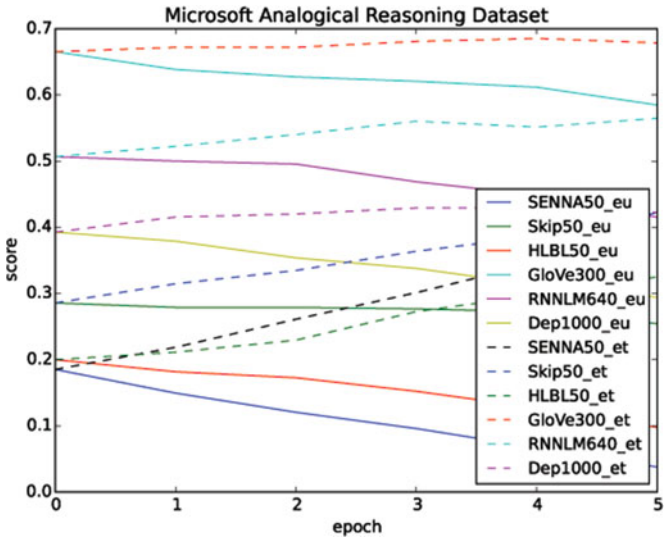


Fig. 6. Comparison of standard SGD and inverse error weighted SGD on Microsoft analogical reasoning dataset.

of the training words rather than the entire vocabulary. For two words, although they are expected to be close to each other, they may move apart after fine tuning because each word is affected by a set of words and the word will change with the majority of the words in the set.

V. Conclusion and Future Direction

In this study, a task independent fine tuning framework has been proposed to boost the performance of word embeddings. The effectiveness of the proposed fine tuning framework has been verified by extensive experiments involving eight datasets and six well known word embeddings. The proposed algorithm for automatic generation and labeling of training data and the inverse error weighted mini-batch SGD algorithm are two pillar factors to the success of the proposed task independent fine tuning framework.

There are still some open problems for this pseudo supervised task independent fine tuning framework. First, some task specific knowledge resources like SenticNet [42] and SentiWordNet [43] can be explored. It would be interesting to see some results on whether the framework will work in task specific fine tuning. Second, the out-of-vocabulary problem deserves more attention. In the experiment, the minimum set of vocabulary of the word embeddings and lexical semantic resources is used, and this results in the out-of-vocabulary problem. In addition, the computational complexity of this fine tuning framework is high. This problem might be addressed by using computationally more efficient nearest neighbour search algorithms.

References

[1] R. E. Banchs, L. F. D'Haro, and H. Li, "Adequacy-fluency metrics: Evaluating MT in the continuous space model framework," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 23, no. 3, pp. 472–482, Mar. 2015.

[2] D. Xiong, M. Zhang, and X. Wang, "Topic-based coherence modeling for statistical machine translation," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 23, no. 3, pp. 483–493, Mar. 2015.

[3] H. Adel, N. T. Vu, K. Kirchhoff, D. Telaar, and T. Schultz, "Syntactic and semantic features for code-switching factored language models," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 23, no. 3, pp. 431–440, Mar. 2015.

[4] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent LSTM neural networks for language modeling," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 23, no. 3, pp. 517–529, Mar. 2015.

[5] G. Mesnil "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 23, no. 3, pp. 530–539, Mar. 2015.

[6] W. Chen, M. Zhang, and Y. Zhang, "Distributed feature representations for dependency parsing," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 23, no. 3, pp. 451–460, Mar. 2015.

[7] S. Clark, "Vector space models of lexical meaning," in *Handbook of Contemporary Semantics, Oxford, U.K.: Blackwell*, 2012.

[8] p. D. Turney, P. Pantel, "From frequency to meaning: Vector space models of semantics," *J. Artif. Intell. Res.*, vol. 37, no. 1, pp. 141–188, 2010.

[9] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

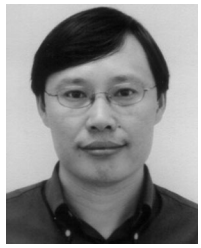
[10] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," in *Proc. 48th Annu. Meeting. Assoc. Comput. Linguistics*, 2010, pp. 384–394.

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv:1301.3781, 2013.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[14] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. 2014 Conf. Empirical Methods Natural Lang. Process*, 2014.
[15] Z. S. Harris, "Distributional structure," *Word*, vol. 14, pp. 3–22, 1954.
[16] J. R. Firth, "A synopsis of linguistic theory 1930–55," in *Studies in Linguistic Analysis. vol. 1952/1959, Oxford, U.K., Philological Soc.*, 1957, pp. 1–32.
[17] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proc. 10th Int. Workshop Artifi. Intell. Statist.*, 2005, pp. 246–252.
[18] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," in *Adv. Neural Inf. Process. Syst.*, 2009, pp. 1081–1088.
[19] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 1751–1758.
[20] A. Passos, V. Kumar, and A. McCallum, "Lexicon infused phrase embeddings for named entity resolution," in *Proc. 18th Conf. Comput. Natural Lang. Learn.*, Baltimore, Maryland, USA, Jun. 26–27, 2014, pp. 78–86.
[21] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *Proc. 52nd Annu. Meet. Assoc. Comput. Linguistics*, 2014, pp. 1555–1565, vol. 1.
[22] M. Bansal, K. Gimpel, and K. Livescu, "Tailoring continuous word representations for dependency parsing," in *Proc. 52nd Annu. Meet. Assoc. Comput. Linguist., Baltimore*, MD, USA, vol. 2, Jun. 22–27, 2014, pp. 809–815.
[23] Z. Chen *et al.*, "Revisiting word embedding for contrasting meaning," in *Proc. ACL*, 2015.
[24] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," arXiv: 1411.4166, 2014.
[25] W.-t. Yih, G. Zweig, and J. C. Platt, "Polarity inducing latent semantic analysis," in *Proc. 2012 Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, Association for Computational Linguistics, 2012, pp. 1212–1222.
[26] C. Xu "Rc-net: A general framework for incorporating knowledge into word representations," in *Proc. 23rd ACM Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA: ACM, 2014, pp. 1219–1228.
[27] S. Qiu, Q. Cui, J. Bian, B. Gao, and T.-Y. Liu, "Co-learning of word representations and morpheme representations," in *Proc. 25th Int. Conf. Comput. Linguistics, Techn. Papers. Dublin City Univ. and Association for Computational Linguistics*, 2014, pp. 141–150.
[28] J. A. Botha and P. Blunsom, "Compositional morphology for word representations and language modelling," in *Proc. 31st Int. Conf. Mach. Learn.*, Beijing, China, Jun. 2014.
[29] G. E. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 833–840.
[30] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[31] D. Lin, "An information-theoretic definition of similarity," in *Proc. 15th Int. Conf. Mach. Learn.*, Madison, Wisconsin, USA, 1998, pp. 296–304.
[32] L. Finkelstein "Placing search in context: The concept revisited," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 406–414.
[33] H. Rubenstein and J. B. Goodenough, "Contextual correlates of synonymy," *Commun. ACM*, vol. 8, no. 10, pp. 627–633, Oct. 1965.
[34] E. Bruni, N.-K. Tran, and M. Baroni, "Multimodal distributional semantics." *J. Artif. Intell. Res.*, vol. 49, pp. 1–47, 2014.
[35] G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren, "Large-scale learning of word relatedness with constraints," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2012, pp. 1406–1414.
[36] D. Yang and D. M. W. Powers, "Verb similarity on the taxonomy of wordnet," in *Proc. 3rd Int. WordNet Conf.*, Jeju Island, South Korea, 2006.
[37] T. Mikolov, W. tau Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proc. 2013 Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.. Association for Computational Linguistics*, May 2013.
[38] G. Zweig and C. J. C. Burges, "A challenge set for advancing language modeling," in *Proc. NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Lang. Model. for HLT*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 29–36.
[39] A. Mnih and G. Hinton, "A scalable hierarchical distributed language model," in *Adv. Neural Inf. Process. Syst.*, vol. 21, 2009, pp. 1081–1088.
[40] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2011, pp. 5528–5531.
[41] A. Fyshe, P. Talukdar, B. Murphy, and T. Mitchell, "Documents and dependencies: an exploration of vector space models for semantic composition," in *Proc. 17th Conf. Comput. Natural Lang. Learn., Association for Computational Linguistics*, 2013, pp. 84–93.
[42] E. Cambria, R. Speer, C. Havasi, and A. Hussain, "Senticnet: A publicly available semantic resource for opinion mining," in *Proc. AAAI Fall Symp., Commonsense Knowl.*, vol. 10, 2010, p. 2.
[43] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in *Proc. 7th Conf. Int. Lang. Resources Eval.*, vol. 10, 2010, pp. 2200–2204.

Xuefeng Yang (M'15) received the Bachelor's degree from Harbin Institute of Technology, Harbin, China, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2011 and 2016, respectively. His Ph.D. thesis is about the vector representation for natural language text. His research interests include applying machine learning models to address semantic representation and semantic matching problems.



Kezhi Mao (M'10) received the B.Eng. degree from Jinan University, Jinan, China, in 1989, the M.Eng. degree from Northeastern University, Shenyang, China, in 1992, and the Ph.D. degree from the University of Sheffield, Sheffield, U.K., in 1998. He was a Lecturer at Northeastern University from March 1992 to May 1995, a Research Associate at the University of Sheffield from April 1998 to September 1998, a Research Fellow at Nanyang Technological University, Singapore, from September 1998 to May 2001, and an Assistant Professor in the School of Electrical and Electronic Engineering, Nanyang Technological University, from June 2001 to September 2005. Since October 2005, he has been an Associate Professor. His research interests include computational intelligence, pattern recognition, knowledge extraction, and big data and text analytics.