

A Boosted Supervised Semantic Indexing for Reranking

Takuya Makino^(✉) and Tomoya Iwakura

Fujitsu Laboratories Ltd., Kawasaki, Japan
{makino.takuya,iwakura.tomoya}@jp.fujitsu.com

Abstract. This paper proposes a word embedding-based reranking algorithm with a boosting. The algorithm converts queries and documents into sets of word embeddings represented by vectors and reranks documents according to a similarity defined with the word embeddings as in Latent Semantic Indexing (LSI) and Supervised Semantic Indexing (SSI). Compared with LSI and SSI, our method uses top- n irrelevant documents of a relevant document of each query for training a reranking model. Furthermore, we also propose application of a boosting to the reranking model. Our method uses the weights of training samples decided by AdaBoost as coefficients for updating model, therefore, highly weighted samples are aggressively learned. We evaluate the proposed method with datasets created from English and Japanese Wikipedia respectively. The experimental results show that our method achieves better mean average precision than LSI and SSI.

1 Introduction

Learning to rank is a machine learning-based method for ranking of information retrieval. With a given training data, learning to rank methods learn weights of features such as rankings given by search engines and content similarities between a query and a document. Documents are ranked based on the scores given by the learned model. One of the approaches of learning to rank uses hand-crafted features [8, 12], such as tf-idf, the title, URL, PageRank and other information. Another approach is a way of learning a model by considering pairs of words between the two texts. The difficulty is that such feature spaces are very large. In order to deal with the difficulty, Supervised Semantic Indexing (SSI) [1] was proposed. For training a ranker, SSI uses a pairwise learning to rank approach that maps a word into a low dimensional vector, which is called as a word embedding, for calculating score. By mapping words into word embeddings, SSI can solve the feature space problem. The SSI approach is suitable for queries represented by natural language used in systems such as an FAQ search and patent searches.

However, the training of SSI does not meet the setting of reranking. The original SSI only handles a negative document for each relevant document of each query (Fig. 1a). In other words, the training setting is different from reranking of top- n search engine results.

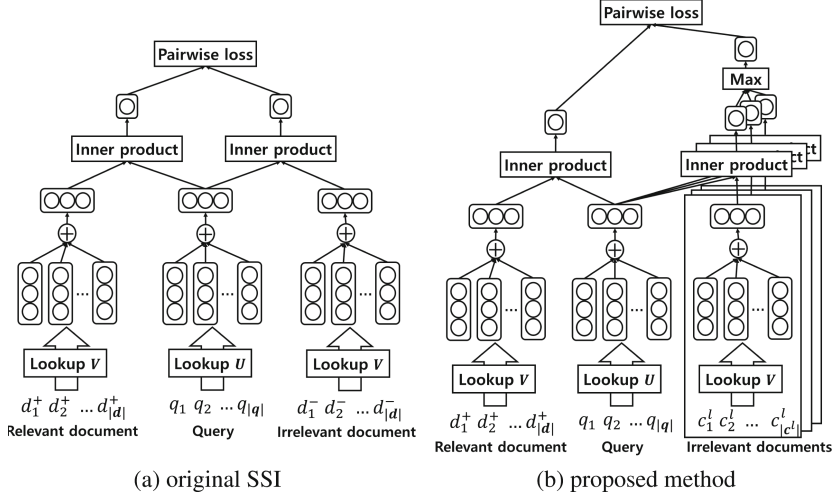


Fig. 1. Overviews of a training of original SSI and our reranking model. \oplus is an element wise addition of vectors. \mathbf{V} is common for relevant and irrelevant documents.

In order to improve the SSI for reranking, we propose the following:

- A training of SSI with irrelevant documents for each relevant document of each query. In our setting, a relevant document and its irrelevant documents are used as a training sample and the training aims at learning a model that gives the highest score to relevant document of each training sample (Fig. 1b).
- Application of a boosting: In addition, in order to get further improved ranking accuracy, we also propose an application of an AdaBoost, which is a boosting algorithm, to the proposed SSI for reranking. Our method uses the weights of training samples decided by the AdaBoost as coefficients for updating model. Therefore, training samples with high weights given by the AdaBoost are aggressively learned.

We first introduce SSI in Sect. 2, then, we propose Reranking SSI with a boosting in Sect. 3. We report the evaluation results of the proposed method with data sets that are created from Japanese and English Wikipedia in Sect. 4. The experimental results show that our method achieves better mean average precision than SSI.

2 Supervised Semantic Indexing

2.1 Preliminary

Equation (1) is a naive score function of SSI, which calculates a similarity between a query $\mathbf{q} \in \mathbb{R}^{N \times 1}$ and a document $\mathbf{d} \in \mathbb{R}^{N \times 1}$.

$$f(\mathbf{q}, \mathbf{d}) = \mathbf{q}^\top \mathbf{W} \mathbf{d}, \quad (1)$$

N is the size of a vocabulary, which is a set of words, and $\mathbf{W} \in \mathbb{R}^{N \times N}$ is a weight matrix of a pair of words in a query and a document. q_i is the value of i -th word in a vocabulary for a query. d_j is the value of j -th word in a vocabulary for a document. The values are such as frequency and tfidf. W_{ij} is the weight of the pair of i -th word and j -th word in a vocabulary.

This function calculates relationship between words in a query and a document. However, there are some problems such as data sparseness and memory requirements (\mathbf{W} requires $O(N^2)$ space). In order to deal with the space problem, Bai et al. [1] proposed to decompose \mathbf{W} into low dimensional matrices.

$$\mathbf{W} = \mathbf{U}^\top \mathbf{V} + \mathbf{I}, \quad (2)$$

where $\mathbf{U} \in \mathbb{R}^{K \times N}$, $\mathbf{V} \in \mathbb{R}^{K \times N}$ and $\mathbf{I} \in \mathbb{R}^{N \times N}$. \mathbf{I} is an identity matrix and K is a dimension of a vector. In other words, \mathbf{U} consists of K -dimensional vectors of words in a query and \mathbf{V} consists of K -dimensional vectors of words in a document. \mathbf{U} and \mathbf{V} can be viewed as a matrix that consists of word embeddings and a word is mapped to a word embedding by looking up a corresponding column vector of those matrices. Equation (1) can be rewritten as follows:

$$\begin{aligned} f(\mathbf{q}, \mathbf{d}) &= \mathbf{q}^\top (\mathbf{U}^\top \mathbf{V} + \mathbf{I}) \mathbf{d}, \\ &= \left(\sum_i^N q_i \mathbf{u}_i \right) \cdot \left(\sum_i^N d_i \mathbf{v}_i \right) + \sum_i^N q_i d_i, \end{aligned}$$

where \mathbf{u}_i is the i -th column vector of \mathbf{U} and \mathbf{v}_i is the i -th column vector of \mathbf{V} . SSI calculates a score that is a summation of two terms. The first term $(\sum_i^N q_i \mathbf{u}_i) \cdot (\sum_i^N d_i \mathbf{v}_i)$ is the inner product between the vector of a query and the vector of a document in a low dimensional space. The vectors of a query and a document are the summations of word embeddings of a query and a document weighted by the value of a corresponding word such as tf-idf. The second term $\sum_i^N q_i d_i$ is the surface similarity, which is the inner product between \mathbf{q} and \mathbf{d} .

For a given query, documents are sorted in a descending order according to scores that are calculated by a model.

2.2 A Training Method

Let $\mathcal{R} = \{(\mathbf{q}_i, \mathbf{d}_i^+, \mathbf{d}_i^-)\}_{i=1}^m$ be a set of training samples where i -th training sample is a tuple of query \mathbf{q}_i , a relevant document \mathbf{d}_i^+ and an irrelevant document \mathbf{d}_i^- . The purpose of training SSI is deriving \mathbf{U} and \mathbf{V} that minimizes the following on \mathcal{R} :

$$\sum_{(\mathbf{q}_i, \mathbf{d}_i^+, \mathbf{d}_i^-) \in \mathcal{R}} \max(0, 1 - f(\mathbf{q}_i, \mathbf{d}_i^+) + f(\mathbf{q}_i, \mathbf{d}_i^-)).$$

Algorithm 1 shows a pseudo code for the training algorithm. At first, \mathbf{U} and \mathbf{V} are initialized by normal distribution with mean zero and standard deviation one.

```

#Training data:  $\mathcal{R} = \{(\mathbf{q}_i, \mathbf{d}_i^+, \mathbf{d}_i^-)\}_{i=1}^m$ 
#The learning rate:  $\lambda$ 
#The maximum iteration of SSI:  $P$ 
Initialize:  $\mathbf{U}, \mathbf{V}$ 
 $p = 1$  while  $p \leq P$  do
  for  $i = 1 \dots m$  do
    if  $1 - f(\mathbf{q}_i, \mathbf{d}_i^+) + f(\mathbf{q}_i, \mathbf{d}_i^-) > 0$  then
      Update  $\mathbf{U}$  by Eq. (3)
      Update  $\mathbf{V}$  by Eq. (4)
    end
  end
   $p++$ 
end
Return  $\mathbf{U}, \mathbf{V}$ 

```

Algorithm 1. A Supervised Semantic Indexing [1].

Then SSI picks up a training sample from \mathcal{R} . If SSI gives a higher score to the pair of a query and an irrelevant document than the pair of the query and a relevant document, \mathbf{U} and \mathbf{V} are updated with Eqs. (3) and (4):

$$\mathbf{U} = \mathbf{U} + \lambda \mathbf{V}(\mathbf{d}^+ - \mathbf{d}^-) \mathbf{q}^\top, \quad (3)$$

$$\mathbf{V} = \mathbf{V} + \lambda \mathbf{U}(\mathbf{d}^+ - \mathbf{d}^-) \mathbf{q}^\top, \quad (4)$$

where λ is a learning ratio.

3 Boosted Supervised Semantic Indexing for Reranking

In this section, we describe the training algorithm of SSI for reranking (ReSSI). Then, we describe an application of a boosting to ReSSI (Boosted ReSSI).

3.1 Extension of SSI to Reranking

Algorithm 2 shows a pseudo code for the training algorithm of ReSSI and Fig. 1 shows an overview of ReSSI. Let \mathcal{R}' be a set of training samples and each sample is a tuple of query \mathbf{q} , a relevant document \mathbf{d}^+ and the set of irrelevant documents $\{\mathbf{c}^l\}_{l=1}^L$ where $\mathbf{c}^l \in \mathbb{R}^{N \times 1}$.

ReSSI learns a model to rank the relevant document \mathbf{d}^+ of each query \mathbf{q} higher than its corresponding irrelevant documents $\{\mathbf{c}^l\}_{l=1}^L$.

In order to use L documents as negative samples for a query, we propose a training method for reranking. ReSSI uses the irrelevant document \mathbf{c}_i^l that has the highest score $f(\mathbf{q}_i, \mathbf{c}_i^l)$ as \mathbf{c}_i^- in the irrelevant documents $\{\mathbf{c}_i^l\}_{l=1}^L$ of query \mathbf{q}_i . If $(1 - f(\mathbf{q}_i, \mathbf{d}_i^+) + f(\mathbf{q}_i, \mathbf{c}_i^-) > 0)$ is satisfied, a current model is updated by Eqs. (5) and (6):

$$\mathbf{U} = \mathbf{U} + \lambda \epsilon_i \mathbf{V}(\mathbf{d}_i^+ - \mathbf{c}_i^-) \mathbf{q}_i^\top, \quad (5)$$

$$\mathbf{V} = \mathbf{V} + \lambda \epsilon_i \mathbf{U}(\mathbf{d}_i^+ - \mathbf{c}_i^-) \mathbf{q}_i^\top, \quad (6)$$

```

#Training data:  $\mathcal{R}' = \{(\mathbf{q}_i, \mathbf{d}_i^+, \{\mathbf{c}_i^l\}_{l=1}^L)\}_{i=1}^m$ 
#The maximum iteration of SSI:  $P$ 
#The weights of samples:  $\{\epsilon_i\}_{i=1}^m$ 
#The learning ratio:  $\lambda$ 
 $\mathcal{R}' = \{(\mathbf{q}_i, \mathbf{d}_i^+, \{\mathbf{c}_i^l\}_{l=1}^L)\}_{i=1}^m$ 
Initialize:  $\mathbf{U}, \mathbf{V}$ 
 $p = 1$ 
while  $p \leq P$  do
  for  $i = 1 \dots m$  do
     $\mathbf{c}_i^- = \arg \max_{\mathbf{c}_i^l} f(\mathbf{q}_i, \mathbf{c}_i^l)$  if  $1 - f(\mathbf{q}_i, \mathbf{d}_i^+) + f(\mathbf{q}_i, \mathbf{c}_i^-) > 0$  then
      Update  $\mathbf{U}$  by Eq. (5)
      Update  $\mathbf{V}$  by Eq. (6)
    end
  end
   $p++$ 
end
Return  $\mathbf{U}, \mathbf{V}$ 
Algorithm 2. A Supervised Semantic Indexing for reranking: ReSSI
( $\mathcal{R}', P, \{\epsilon_i\}_{i=1}^m, \lambda$ )

```

where $\{\epsilon_i\}_{i=1}^m$ are the weights of training samples. We set $\epsilon_i = 1$ ($1 \leq i \leq m$) for ReSSI without boosting. If we use $L = 1$ and $\epsilon_i = 1$ ($1 \leq i \leq m$), this is equivalent to the original SSI.

3.2 Application of Boosting

We apply a variant of AdaBoost [9] to ReSSI. The original AdaBoost was designed for the classification problem. Here, we propose to apply the method designed for structured prediction [11] to a learning to rank.

We show a pseudo code of the application of the boosting to ReSSI in Algorithm 3. In the first iteration, ReSSI is trained with the initial weights of samples $\epsilon_i = 1/m$ ($1 \leq i \leq m$).

Then, the boosting learner updates the weights of training samples. The boosting learner assigns larger weights to training samples that are incorrectly ranked. To realize this, we define a loss for a training sample $(\mathbf{q}_i, \mathbf{d}_i^+, \{\mathbf{c}_i^l\}_{l=1}^L)$ as follows:

$$s_t(\mathbf{q}_i, \mathbf{d}_i^+, \{\mathbf{c}_i^l\}_{l=1}^L) = f_t(\mathbf{q}_i, \mathbf{d}_i^+) - f_t(\mathbf{q}_i, \mathbf{c}_i^-),$$

where, $\mathbf{c}_i^- = \arg \max_{\mathbf{c}_i^l} f_t(\mathbf{q}_i, \mathbf{c}_i^l)$ and $f_t(\mathbf{q}_i, \mathbf{c}_i^l) = \mathbf{q}_i^\top (\mathbf{U}_t^\top \mathbf{V}_t + \mathbf{I}) \mathbf{c}_i^l$.

After the boosting learner learns a weak learner at time t , it searches a confidence-value α_t that satisfies $\tilde{Z}_t(\tilde{\alpha}_t) < 1$:

$$\tilde{Z}_t(\tilde{\alpha}_t) = \sum_{i=1}^m w_{t,i} e^{-\tilde{\alpha}_t s_t(\mathbf{q}_i, \mathbf{d}_i^+, \{\mathbf{c}_i^l\}_{l=1}^L)},$$

```

#Training data:  $\mathcal{R}' = \{(\mathbf{q}_i, \mathbf{d}_i^+, \{\mathbf{c}_i^l\}_{l=1}^L)\}_{i=1}^m$ 
#The iteration of SSI training:  $P$ 
#The learning ratio of SSI training:  $\lambda$ 
Initialize:  $U = \{\}, V = \{\}, A = \{\}$ 
 $t = 1$ 
set initial value:  $w_{1,i} = \frac{1}{m}$  (for  $1 \leq i \leq m$ )
while  $t \leq T$  do
     $\mathbf{U}_t, \mathbf{V}_t = \text{ReSSI}(\mathcal{R}', P, \{w_{t,i}\}_{i=1}^m, \lambda)$ 
    Find  $\tilde{\alpha}_t$  that satisfies  $\tilde{Z}_t(\tilde{\alpha}_t) < 1$ .
     $U \leftarrow U \cup \{\mathbf{U}_t\}$ 
     $V \leftarrow V \cup \{\mathbf{V}_t\}$ 
     $A \leftarrow A \cup \{\tilde{\alpha}_t\}$ 
    for  $i = 1 \dots m$  do
        | Update sample weights by Eq. (7)
    end
     $t++$ 
end
Return  $U, V, A$ 

```

Algorithm 3. A Boosted Supervised Semantic Indexing

where $w_{t,i}$ is the weight of i -th sample at time t and e is Napier's constant. We find $\tilde{\alpha}_t$ with the same method used in [11].

After obtaining α_t , the weight of each sample is updated as follows:

$$w_{t+1,i} = w_{t,i} \frac{e^{-\alpha_t s_t(\mathbf{q}_i, \mathbf{d}_i^+, \{\mathbf{c}_i^l\}_{l=1}^L)}}{\tilde{Z}_t(\alpha_t)}. \quad (7)$$

After training a boosting learner, we obtain t ReSSI models and their confidence values. Given a query and a document, a final ranker calculates scores by using all ReSSI models. The score of a document given by a final ranker is a summation of scores given by ReSSI models.

$$f^*(\mathbf{q}, \mathbf{d}) = \sum_{t=1}^T \tilde{\alpha}_t f_t(\mathbf{q}, \mathbf{d}).$$

4 Experiments

4.1 Dataset

We conduct our experiments on datasets generated from English Wikipedia and Japanese Wikipedia as in [1] with the following steps¹.

¹ We used <https://dumps.wikimedia.org/jawiki/20160407/jawiki-20160407-pages-articles.xml.bz2> and <https://dumps.wikimedia.org/enwiki/20160901/enwiki-20160901-pages-articles.xml.bz2>. Retrieved October 14, 2016.

- (d1) Preprocessing: Before generating data sets, we removed markup from articles in Wikipedia by using regular expressions. For extracting words from English articles, we used white space. For tokenizing Japanese articles, we used MeCab², which is a Japanese morphological analyzer.
- (d2) Selecting Wikipedia articles as queries: 10,000 articles for training and 1,000 articles for test are randomly sampled from each Wikipedia archive as queries.
- (d3) Collecting positive samples of queries: We used Wikipedia articles linked by each query Wikipedia article collected at (d2) as positive samples of the query.
- (d4) Collecting negative samples of queries: We collected negative samples by searching articles in each Wikipedia archive with words in each query article. We used a full text search engine Elasticsearch³. OR search of words in each query article was used for collecting irrelevant articles as negative samples. We collected $L = 10$ articles other than positive documents for each query. In the training phase, for generating training data for a query that has multiple relevant documents, we assign the same irrelevant documents obtained with the query to each relevant document. For evaluation, a set of the relevant documents and the L irrelevant documents of each query is given to a ranking algorithm.

Table 1. Size of training and test data sets.

English		Japanese	
Train	Test	Train	Test
72,816	7,820	243,600	25,358

Table 1 lists the size of training data. Average numbers of words that are in vocabulary of training data and test data are 241 and 465 respectively.

4.2 Methods to Be Compared

The following algorithms are compared with our methods; ReSSI and Boosted ReSSI.

tfidf. Documents are sorted in descending order based on an inner product between a vector of query and a vector of a document. The weight of each word in a query and a document is normalized tfidf. A normalized tfidf is an original tfidf divided by the number of words in the query and the document respectively.

² <https://github.com/taku910/mecab>. Retrieved October 14, 2016.

³ <https://www.elastic.co/downloads/past-releases/elasticsearch-1-7-1>, Retrieved October 14, 2016.

Table 2. Evaluation results on enwiki and jawiki. The scores with * significantly differs with ones of Boosted ReSSI with $p \leq 0.01$. The scores with † significantly differs with ones of ReSSI with $p \leq 0.01$.

	MAP	
	enwiki	jawiki
tfidf	0.016*†	0.011*†
LSI	0.314*†	0.069*†
SSI	0.573*†	0.392*†
ReSSI	0.623*	0.482*
Boosted ReSSI	0.634 †	0.497 †

LSI. Latent Semantic Indexing [6]. LSI is an unsupervised dimensional reducing model. We used SVDLIBC^{4,5} for obtaining latent parameters. Documents are sorted in descending order based on inner product with the vector of a query in a latent space.

SSI Algorithm 1. Original paper randomly selected a negative sample of each relevant document. In this paper, we select a negative sample from the document that has the highest score in a search result obtained with a query of each relevant document.

ReSSI. This is our proposed reranking-based SSI. The negative sample of each query is selected from the top 10 search result of a search engine obtained with the query.

Boosted ReSSI. This is the boosting version of ReSSI.

We set the size of the dictionary to $N = 30,000$, and the maximum iteration number for SSI and ReSSI to $P = 30$. For the proposed method, we set the maximum iteration number for boosting to $T = 5$. Dimensions of word embeddings in SSI, ReSSI and Boosted ReSSI are set to $K = 50$. \mathbf{U} and \mathbf{V} are initialized by normal distribution with mean zero and standard deviation one. The learning ratios of each algorithm is following. For SSI and ReSSI, we used $\lambda = 0.01$. For Boosted ReSSI, we used $\lambda = 0.01 \times m$ this is because the larger m is given, the less weight is given for each training sample by the AdaBoost. We implemented SSI, ReSSI and Boosted ReSSI with C++.

4.3 Evaluation Metric

We use Mean Average Precision (MAP), the mean of average precision over a collection of questions, as our evaluation metric. For evaluation, a set of the r_i relevant documents and the L irrelevant documents of each query \mathbf{q}_i is ranked with a reranking algorithm first. Let $[\mathbf{d}_i^1, \dots, \mathbf{d}_i^{T_i+L}]$ be a list of documents of

⁴ <https://tedlab.mit.edu/~dr/SVDLIBC/>. Retrieved October 14, 2016.

⁵ We used following options: `-d 50 -i 5 -e 1e-30 -a las2 -k 1e-6`.

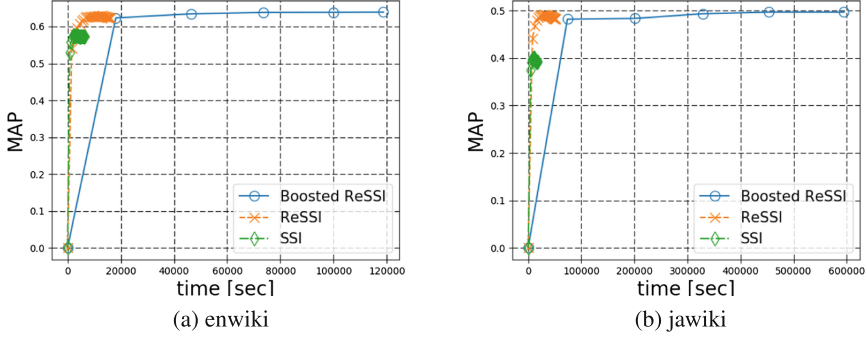


Fig. 2. Accuracy on each training time. MAPs are calculated on test data.

\mathbf{q}_i ranked by a reranking algorithm. \mathbf{d}_i^1 is the top ranked document for \mathbf{q}_i . We calculate MAP as follows:

$$\text{MAP} = \frac{1}{C} \sum_{i=1}^C \frac{\sum_{j=1}^{r_i+L} P_{i,j}}{r_i},$$

where, C is the size of test data, $P_{i,j}$ is the precision if the j -th document of \mathbf{q}_i is relevant, otherwise zero. $P_{i,j}$ is calculated as follows:

$$P_{i,j} = \frac{R_j}{j} [[\mathbf{d}_i^j \text{ is a relevant document}]],$$

where R_j is the number of relevant documents in the top- j documents and $[[\pi]]$ is 1 if a proposition π holds and 0 otherwise.

$\frac{\sum_{j=1}^{r_i+L} P_{i,j}}{r_i}$ is 1 if all r_i relevant documents of \mathbf{q}_i are ranked in top- r_i for all questions, otherwise it becomes smaller than 1.

4.4 Accuracy

Table 2 shows main result of our experiments on enwiki and jawiki. We see that ReSSI shows higher accuracy than tfidf, LSI and SSI. Furthermore, Boosted ReSSI outperforms all the other methods. For significance test, we used paired t-test. For calculating p-value, we used average precision for each query. There are significant differences between ReSSI and tfidf, LSI and SSI. Furthermore, there are also significant differences between Boosted ReSSI and the others. MAP of tfidf is lower than other methods. This indicates that predicting link relation between a query and a document based on surface word matching similarity is difficult.

4.5 Transition of Accuracy by Training Time

We compared MAPs for each model with different sizes of training data. In Fig. 2, we plot MAP of test data on each epoch in the training phase. We use the seconds of epoch as horizontal axis instead of the number of epochs. For the boosting algorithm, we use the sum of the seconds by each boosting round as horizontal axis. SSI converges faster than ReSSI since SSI fixes negative examples. However, the performance of ReSSI is better than SSI.

4.6 Accuracy on Different Training Data Size

Figure 3 shows the results of different sizes of training data. For both datasets, we can see that our proposed methods show higher accuracy than SSI not only the larger training data set but also smaller training data sets.

4.7 Reranking Speed

We compared the times of reranking of ReSSI and Boosted ReSSI. For measuring times, we used models that are trained on 50% samples that are randomly sampled from enwiki and jawiki respectively. Times are measured on Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40 GHz and we used a single CPU for each model. Table 3 shows the average seconds of predicting of our proposed methods over test queries. The time of reranking of Boosted ReSSI is almost five times, which is the number of boosting round in our experiments, times larger than ReSSI.

For the AdaBoost to structured perceptron [11], merging weak learners by addition is effective. However, for Boosted ReSSI, the approach is not effective. This is because the boosting learner needs to calculate the product of matrices \mathbf{U}_t and \mathbf{V}_t when merging a weak learner by addition and it requires $O(N^2)$ memory complexity and $O(|\mathbf{q}||\mathbf{d}|N)$ computational complexity where $|\mathbf{q}|$ and $|\mathbf{d}|$ are the number of non zero values in \mathbf{q} and \mathbf{d} respectively.

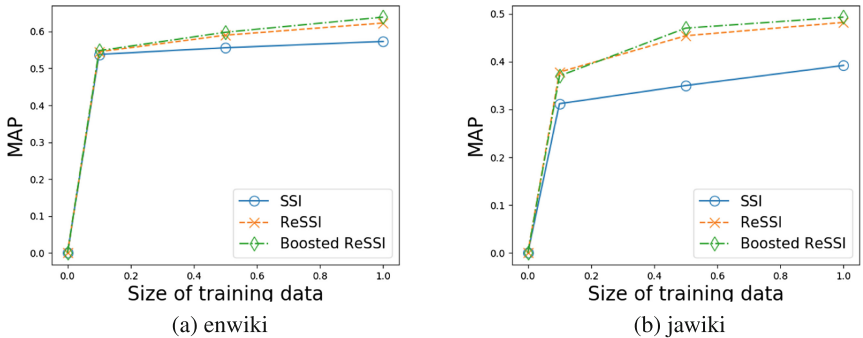


Fig. 3. Transition of Accuracy with different sizes of training data. We randomly sampled 10% and 50% of training data from enwiki and jawiki respectively.

However, if we have multiple CPU or multicores, we can easily improve the reranking speed by distributing the calculation because we can independently calculate scores given by each model created in boosting.

Table 3. Average seconds of reranking

	enwiki	jawiki
ReSSI	0.081	0.234
Boosted ReSSI	0.386	1.101

5 Related Works

For obtaining word vectors, there are ways such as matrix factorization based approach and learning to predict words given context. Latent Semantic Indexing [6], probabilistic Latent Semantic Analysis [2] and Latent Dirichlet Allocation [4] are well-known matrix factorization based vector space models without hand-crafted features. These models can map a word into a low dimensional vector for calculating a score between a query and a document. Skip-gram, CBOW and GloVe [14, 16] learns to predict words given context words. For predicting a word, these methods calculate inner product between a word and its context word. In contrast, our method learns word embeddings and models to rank a relevant document higher than an irrelevant document.

Supervised learning to rank models that map a word into a low dimensional vector are proposed in various tasks such as sentiment classification and image retrieval task [3, 10, 15, 18]. Our focus is training methods of a reranking model for the pair of a query and a document.

For updating a ranking model, Bespalov et al. [3] proposed WARP loss that is based on the rank of a relevant document. Recent works of optimization methods for learning parameters of neural networks modifies a learning ratio based on square of gradient [7, 13, 21]. Our method uses the sample weight decided by a boosting. These methods can be applied to improve our proposed method.

RankBoost [8] and AdaRank [19] are boosting algorithms for ranking problem. Since these methods use a ranking feature or a decision tree as a weak learner, it is not scalable when the dimension of a feature space is large like our experimental setting. AdaMF [17] is similar with our proposed method. This method is an application of boosting to matrix factorization for a recommender system. AdaMF uses a point-wise learning to rank approach that predict the rate of an item by a user. Our method belongs to a pairwise learning to rank approach, such as [5].

ABCNN [20] is a convolution neural network with attention for question answering as a classification rather than reranking and shows high mean average precision. We can integrate attention mechanism into our methods for obtaining higher mean average precision.

6 Conclusion

We proposed a embeddings-based reranking with a boosting. Our embeddings-based reranking model learns to rank a relevant document higher with a similarity defined calculated word embeddings. Experimental results on English and Japanese Wikipedia showed that the proposed method outperformed SSI.

References

1. Bai, B., Weston, J., Grangier, D., Collobert, R., Sadamasa, K., Qi, Y., Chapelle, O., Weinberger, K.: Supervised semantic indexing. In: Proceedings of the 18th CIKM (CIKM 2009), pp. 187–196 (2009)
2. Berger, A., Lafferty, J.: Information retrieval as statistical translation. In: Proceedings of the 22nd SIGIR (SIGIR 1999), pp. 222–229 (1999)
3. Beshalov, D., Bai, B., Qi, Y., Shokoufandeh, A.: Sentiment classification based on supervised latent n-gram analysis. In: Proceedings of the 20th CIKM (CIKM 2011), pp. 375–382 (2011)
4. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
5. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), pp. 89–96 (2005)
6. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.* **41**(6), 391–407 (1990)
7. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Technical report UCB/EECS-2010-24. EECS Department, University of California, Berkeley, March 2010
8. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.* **4**, 933–969 (2003)
9. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **55**(1), 119–139 (1997)
10. Grangier, D., Bengio, S.: A discriminative kernel-based model to rank images from text queries. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)* **30**, 1371–1384 (2008)
11. Iwakura, T.: A boosted semi-markov perceptron. In: Proceedings of the 17th CoNLL, pp. 47–55 (2013)
12. Joachims, T.: Optimizing search engines using click through data. In: Proceedings of the 8th KDD (KDD 2002), pp. 133–142 (2002)
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. *CoRR* abs/1412.6980 (2014)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems*, vol. 26, pp. 3111–3119. Curran Associates, Inc. (2013)
15. Min, K., Zhang, Z., Wright, J., Ma, Y.: Decomposing background topics from keywords by principal component pursuit. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM 2010), pp. 269–278 (2010)

16. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543 (2014)
17. Wang, Y., Sun, H., Zhang, R.: AdaMF: adaptive boosting matrix factorization for recommender system. In: Li, F., Li, G., Hwang, S., Yao, B., Zhang, Z. (eds.) *WAIM 2014*. LNCS, vol. 8485, pp. 43–54. Springer, Cham (2014). doi:[10.1007/978-3-319-08010-9_7](https://doi.org/10.1007/978-3-319-08010-9_7)
18. Weston, J., Bengio, S., Usunier, N.: Large scale image annotation: learning to rank with joint word-image embeddings. In: *European Conference on Machine Learning* (2010)
19. Xu, J., Li, H.: AdaRank: a boosting algorithm for information retrieval. In: *Proceedings of the 30th SIGIR (SIGIR 2007)*, pp. 391–398 (2007)
20. Yin, W., Schütze, H., Xiang, B., Zhou, B.: ABCNN: attention-based convolutional neural network for modeling sentence pairs. *TACL* **4**, 259–272 (2016)
21. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701 (2012)

Information Retrieval Technology

13th Asia Information Retrieval Societies Conference,
AIRS 2017, Jeju Island, South Korea, November 22-24,
2017, Proceedings

Sung, W.-K.; Jung, H.; XU, S.; Chinnasarn, K.; Sumiya, K.;
Lee, J.; Dou, Z.; Yang, G.H.; Ha, Y.-G.; Lee, S. (Eds.)

2017, XII, 235 p. 53 illus., Softcover

ISBN: 978-3-319-70144-8