A Document Query Search
Using an Extended Centrality with the Word2vec

Wooju Kim
Dept. of Information and Industrial Engineering, Yonsei University
50 Yonsei-ro, Seodaemun-gu, Seoul, Korea
wkim@yonsei.ac.kr

Heewon Jang
Dept. of Information and Industrial Engineering, Yonsei University
50 Yonsei-ro, Seodaemun-gu, Seoul, Korea
hwjang1000@gmail.com

Hak-Jin Kim∗
School of Business, Yonsei University
50 Yonsei-ro, Seodaemun-gu, Seoul, Korea
hakjin@yonsei.ac.kr

Donghe Kim
Korea Railroad Research Institute
176 Cheoldobangmulgwan-ro, Uiwang-si, Korea
kdh777@krri.re.kr

ABSTRACT
While everyday document search is done by keyword-based queries to search engines, we have situations that need deep search of documents such as scrutinies of patents, legal documents, and so on. In such cases, using document queries, instead of keyword-based queries, can be more helpful because it exploits more information from the query document. This paper studies a scheme of document search based on document queries. In particular, it uses centrality vectors, instead of tf-idf vectors, to represent query documents, combined with the Word2vec method to capture the semantic similarity in contained words. This scheme improves the performance of document search and provides a way to find documents not only lexically, but semantically close to a query document.

CCS Concepts
• Information systems → Retrieval models and ranking; Similarity measures; • Computing methodologies → Machine learning; • Computer systems organization → Architectures;

Keywords
Word2vec; similarity; neural network language model; document search

∗the corresponding author

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

ICEC’16 August 17-19, 2016, Suwon, Republic of Korea
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-4222-3/16/08 . . . $15.00
DOI: http://dx.doi.org/10.1145/2971603.2971617

1. INTRODUCTION
Document search has been a popular and important topic in the field of information retrieval. How to find the documents that a user wants or that are relevant to what a user wants, has been the key issue of study. The conventional strategy to achieve the goal is to index keywords of documents and to find documents that contains keywords close to the keywords users want. But users sometimes meet the situations requiring deeper search. For example, if a user works as a patent officer, she or he receives many proposals for patents, scrutinizes documents of patents registered already, and should check if the proposals take the contents of any patents registered. If that document is not written in formal terminologies, it is not easy to find relevant documents by using indexed keywords. In such cases, search with a query document can be very useful; a machine calculates weights of importance for not few but all words contained in a query document, and uses them to compare others and to find relevant documents for users. In our study, we consider the situation of document search with a document query.

A frequented representation tool in document search is the vector space model. A document in a pool can be represented with a vector of tf-idf scores; that is, for each word in the document, the product of the term frequency of that word against that document and the inverse document frequency against the document pool becomes the score for the word in the document, and forms the entry in the representation vector for the document. The identification of similar documents is done by the similarity measure such as the cosine similarity. But using this vector space model to represent a query document raises some limitations. First, a pair of longer documents may have very low similarity, because they are less likely to have the same words and the inverse document frequency against the document pool becomes the score for the word in the document, and forms the entry in the representation vector for the document. The identification of similar documents is done by the similarity measure such as the cosine similarity. But using this vector space model to represent a query document raises some limitations. First, a pair of longer documents may have very low similarity, because they are less likely to have common positive entries in their vectors. Second, this modeling scheme subsumes only the lexically same words in the similarity calculation. Finally, the model also ignores the sequential order of occurrences of multiple words, even though the layout of words can give unshammable information about the context.

Particularly, the second limitation impairs search capac-
ity severely. When a query document is expressed in different terminologies other than jargons, the limitation can fail search for the similar documents. Figure 1 explains it with an example. It considers two documents involved: one contains three words “Cars”, “run” and “fast”, and the other “Autos”, “move” and “fast”. Then it assumes that their document vectors are \((3, 0, 2, 0, 1)\) and \((0, 2, 0, 3, 1)\) for the involved words, respectively. In this example, even though the meanings of those documents are very similar—“cars” and “run” are similar to “autos” and “move”—the cosine similarity score is mere 0.07 because their vectors share only one positive value in their scores. The semantic similarity of the two documents is not captured by the vector space representation. Using the Word2vec model would remedy this limitation. Also it may overcomes the other two limitations. Including semantically close words into consideration may make more positive scores in semantically similar words. The Word2vec is using a distributive representation model which naturally takes layouts of words in context into account.

In addition, representing a query document with the vector space model is not a good idea in the viewpoint of computation. Calculating tf-idf scores for all words in a query document against all documents in a pool is computationally time consuming, and accrued lags from a sequential document queries makes unpleasant user experience. For a query document, hence a different way of representation is needed. This can be eased by using simple occurrence vectors or centrality vectors by extracting word information within the query document.

This research therefore proposes a novel method, which overcomes those limitations, to represent a query document and to measure its similarity to documents in a pool. The fundamental idea is to use the centrality scores for building representation vectors for a query documents, and to resort to a neural network language model, the Word2vec model, in order to improve the performance of search by considering semantic similarities in words.

This paper is organized as follows. After this section, some related works are reviewed, and Section 3 explains the document similarity scheme proposed in this paper. Performance evaluation next follows in Section 4, and finally it ends with the conclusion in Section 5.

2. RELATED WORK

This study, to represent a document in a document pool, uses tf-idf document vectors. The vector space model for representation has long history in the field of information retrieval since 1970s. [14, 8], when a collection of documents is given, found a way to index documents automatically and to make their comparison easy. In particular, [14] introduced a measure of weighing a term according to the frequency of its occurrence in a document \((tf)\) with a factor inversely related to the document frequency with respect to the term \((idf)\). When a document considered as a sequence of terms or words, a document can be represented as a vector of those scores about the terms forming the document. This scheme is still used in the field of the natural language processing to display the importance of a word and thus to represent a sequence of words as a vector.

The centrality is a concept from the field of graph theory or network analysis to identify important vertices in a given network. This concept became critical in information retrieval because in the era of the Internet a search engine, based on that concept, became a must-have to retrieve so much information disseminated in a big network, the Internet, together with the development of social media. Since graph is so versatile and applied to various fields, the literature on centrality is so vast. This research uses PageRank which is an eigenvalue centrality. PageRank is the first algorithm used by Google for its search engine, and is described in details in [6]. An open source library, Gephi [1], provides an implementation of PageRank as its component, which is used in this study. [10] applied the PageRank idea to ranking documents not linked by hypertexts, by generating links by a language model. [9] proposed another eigenvalue centrality for text summarizing, an issue of the natural language processing.

Recently the neural network approach is used in building stochastic language models. [3, 2, 4] ushers the neural network to the field of language modeling. Based on the insight of n-grams, it contrives a way to make a distributed representation for words. The connections of a word and its neighboring words are formulated as a probabilistic model that makes a two-layer neural network model. In that paper, the authors propose a parallel version of the back propagation algorithm that computes feature vectors of words in a given vocabulary. Even though the neural network approach seems superior to the conventional n-gram method, it is known that its training requires bigger size of corpus and takes much more time. The subsequent studies naturally move the focus on the faster implementation in training, and [5, 13] propose hierarchical distributed language models by using the concept of binary tree. The most practical neural language models are ones proposed by [11, 12]. The conventional natural language processing techniques have been simple and atomic. It is because the simplicity may enhance flexibility in constructing state-of-art systems and improving extant systems. But as simple techniques reach limits to achieve no progress, more advance techniques have attention. It happens in language modeling, and neural network techniques outperform simple n-gram models. [11] proposes two simple log-linear models: continuous Bag-of-Words model and continuous Skip-gram model. Its models, different from [4], build simple structure by removing the non-linear hidden layer. The former has context words as its inputs and predicts the current word while the latter has the current word as its input and output context words. Comparing with the previous techniques, those models improves accuracy at much lower computational cost; in particular, for measuring syntactic and semantic word similarities. [12] claims the Skip-gram model as a tool of learning distributed vector representation. This representation importantly catches hold of relationship of words syntactically and semantically. Because of this property, this research exploits this model as a component of a system that measures document similarities. Also the paper shows some improving techniques in reducing the time of model training with a publication of the Word2vec code. An implementation based on that code is used in the research. The idea of using the neural network in the natural language processing changes the study in a holistic way. For example, [7] suggests a unified framework of conducting multiple tasks in natural language processing by a deep neural network model such as part of speech tagging, named entity recognition, semantic role labeling, language models and semantically related...
3. A DOCUMENT SIMILARITY SCHEME

3.1 Query graph

Given a query document \( q \), we can construct a query graph with all the words in the document, denoted by \( G_q = (V_q, qw) \). The set \( V_q \) of all nodes in \( G_q \) consists of all the words in \( q \). Edges on the graph can be constructed by the weight function \( qw(u, v) \) for two words \( u \) and \( v \). In this paper, we use the point-wise mutual index (PMI) for the function,

\[
qw(u, v) := \text{pmi}(u, v) = \log \frac{p(u, v)}{p(u)p(v)}
\]

where \( p(x) \) is the probability of occurrence of word \( x \), and \( p(x, y) \) the probability of co-occurrence of words \( x \) and \( y \) in the document \( q \). They are approximated by counting the occurrence and the co-occurrence in the document. Under the functional values, each edge is constructed only if \( qw > qw_{\text{min}} \) with some threshold value \( qw_{\text{min}} \).

3.2 Centrality by the PageRank

The query document is represented with a vector according to the framework of the vector space model. The query document is formed with its containing words, and it can be represented as a vector of scores corresponding to the words. In our graph \( G_q \), words make the set of nodes and the edges are based on the PMI scores. The higher score manifests collocation of two relevant words, and a word with more links to other words are closely related to others. Such a word may play a critical role to represent the content of the document. Based on the theory of centrality on network, we can hence premise that a word of the higher centrality score may be more important and describe significantly the core idea of the document. In this paper, we capture the importance of words with a variant of the eigenvector centrality, the PageRank [6] which is defined as:

\[
C(t + 1) = (1 - d) \frac{1}{N} 1 + dC(t) \tilde{A}
\]

where \( N \) is the number of nodes in \( G_q \), \( C \) the row vector of centrality score for each node, \( 1 \) the row vector whose entries are all one, and \( \tilde{A} \) the adjacent matrix of \( G_q \) normalized with each column sum one. [6] uses a parameter \( d \) called the damping factor. Any node’s rank is determined by those of other linked nodes and as the value of \( d \) decreases, this effect is damped by the first term.

3.3 Representation of documents in a pool

To search documents similar to a given query document, one usually applies the cosine similarity to measure the similarity score between them. For this purpose, all documents should be represented by a vector according to the vector space model. The tf-idf measure works for this role. For the corpus of all words from a pool of documents, the term frequencies for each document are calculated and weighted by the inverse document frequencies to give more weight to rare terms. This tf-idf score vector is recorded as a representative of the given document, and a pair of tf-idf score vectors are used in the calculation of the cosine similarity which gives the extent of similarity of the corresponding documents. Now the centrality vector of the given query document can be used to search similar documents from a pool of documents in place of the tf-idf vector in the cosine similarity calculation. As mentioned earlier, however, the simple use of the centrality vector in calculating the cosine similarity can ignore documents of the same content with different words because the centrality cannot capture semantics of words and lexically different words can be differentiated even if with the same meaning. In Figure 1 the first document has the centrality vector whose cosine similarity with the other is really low (0.07). The first does not use the words “autos” and “move” in its description while the other does not “cars” and “run”. That is why they have only one common positive entry in the vectors in spite of the fact that “cars” and “autos” have the same meaning as well as “run” and “move”.

3.4 The Word2vec model

The Word2vec method was proposed to compute continuous vector representations of words from corpora [11, 12]. What makes this approach differentiated from the previous works is that it reflects the word-similarity implicitly in representations of words. Our scheme of document similarity requires a representation of a document, and the integration of the idea of the word-similarity into the document representation may improve the performance of computing the document similarity in document search. This approach of the artificial neural networks uses a model called the Skip-gram model [12]. Under the given training words \( w_1, \ldots, w_T \), its objective is to maximize over the parameter \( \theta \) the following log-linear expression:

\[
\max_{\theta} \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t; \theta)
\]
where $c$ is the size of the training context. By the training, the best estimator for $\theta$ and the feature vectors for the words are computed as the output. The similarity between two words is finally calculated by their feature vectors. Such similarity scores for pairs of words form a matrix as in Table 1. The table shows the similarity matrix for the example given in Figure 1.

### 3.5 Extended centrality model

After obtaining the word similarity matrix by the Word2vec neural network method, the proposed extended centrality for a document is defined by the matrix multiplication of the conventional centrality score vector and the matrix as follows:

\[
\mathcal{W} \quad \text{The set of all words appeared in all documents, i.e. the corpus} \\
K_j \quad \text{The set of all words most similar to the word } j \\
C_{ij} \quad \text{The centrality score for the word } j \text{ in the document } i \\
\tilde{C}_{ij} \quad \text{The extended centrality score for the word } j \text{ in the document } i \text{ using the Word2vec method} \\
C_i \quad \text{The centrality score vector for the document } i \text{ whose entries are } C_{ij} \\
\hat{C}_i \quad \text{The extended centrality score vector for the document } i \text{ whose entries are } \tilde{C}_{ij}, \text{ using the Word2vec method} \\
s_{kj} \quad \text{The Word2vec similarity score between the words } k \text{ and } j \\
M \quad \text{The similarity matrix consisting of } s_{kj} \text{ with its dimension } |W| \times |W| \\
\tilde{C}_i = C_i M \\
\hat{C}_{ij} = C_{ij} + \sum_{k \in K_j, k \neq j} C_{ik} s_{kj} \quad \forall j \in \mathcal{W}
\]

The equations (4) and (5) show the measure for a given document $i$. In the right-hand side of the equation (5), the first term is exactly the centrality score of the word $j$ in the document $i$ because the entry of the matrix between the word $j$ and itself is always one.

From the example of Figure 1, we have the Word2vec similarity matrix in Table 1. It says that each word has score one with itself at the diagonals, and the pair of “cars” and “autos” has 0.7 and the other of “run” and “move” 0.5 as their similarity scores. The matrix multiplication of the centrality vector for the first document in Figure 1, $\{3, 0, 2, 0, 1\}$, with the matrix of Table 1 results in the modified centrality vector for the first document in Figure 1, $\{0.7, 1, 0, 0, 0\}$. The cosine similarity scores with the vector for the other document, $\{0, 2, 0, 3, 1\}$, are 0.07 and 0.50. The shaded regions in Figure 2 have more positive entries than that of Figure 1 because the similarity information of words is integrated in the first vector. Clearly the two documents are found out to be more similar in the measure of Figure 2. This fact reflects the reality more suitably since their contents are the same but only use different synonyms.

The procedure of document search with a document query is summed up in Figure 3. Each document in the document repository is represented with a tf-idf vector, and the learning process runs over the repository and determines the parameters of the Word2vec model. A word similarity matrix is obtained from the determined Word2vec model. When a user queries the system with a query document, its centrality vector is calculated on the word graph made by the PMI scores. The inner product of the vector and the similarity matrix brings out the query document vector $x_q$. The similarity of two documents, a document $d_i$ in the pool and the query document $d_q$, is finally calculated with their representatives $x_i$ and $x_q$, which is used in displaying the search result; i.e. as a result the system displays the documents above a given fixed threshold value in their document similarity scores.

### 4. EVALUATION

In this section, A computational experiment is conducted to evaluate the performance of the proposed method. The experiment is done with a R&D proposal from Korea Railroad Research Institute (KRRI) as a query document; one about Bimodal Transportation System (BTS). After KRRI, as a research institute about laws, orders, and regulations for transportation systems, receiving research a proposal, it needs to review related law and order documents describing the issues related to the proposed research subjects in the evaluation of proposals. This process necessarily requires to search all related documents out of a given legal document pool.

4485 legal documents are initially collected from the law and order information system to form a pool of documents, and the morphological analysis is conducted over the pool to make tf-idf representation vectors for all collected documents. Also, the morphological analysis is conducted on the proposal to obtain the information of words contained. Out of the collected documents, the ones related to the subjects of the proposal are screened out. BTS has strong relationship with 78 documents and weak relationship with 8 documents. In this pre-process, it is found out that 22 laws and orders are related to BTS.

In the calculation of the Word2vec representation vector for a given query document, a word similarity matrix should be constructed. It commands the construction of feature vectors by learning the Word2vec model. This research uses Python gensim topic modeling library\(^1\) that contains a C-language implementation\(^2\) of the Word2vec model\(^3\), and its learning process is done on the all collected legal documents. [11] suggested two log-linear neural network language models, the continuous Bag-of-Words model and the continuous Skip-gram model, for estimated word representation, but later [12] recommended for the distributed vector representation that captures the syntactic and semantic word relationships between relevant words. So this research uses the

\(^1\)https://radimrehurek.com/gensim/index.html
\(^2\)The code can be downloaded from http://word2vec.googlecode.com/svn/trunk/.
\(^3\)https://code.google.com/p/word2vec

<table>
<thead>
<tr>
<th>Table 1: An example for the Word2vec similarity matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>cars autos run move fast</td>
</tr>
<tr>
<td>cars 1 0.7 0 0 0</td>
</tr>
<tr>
<td>autos 0.7 1 0 0 0</td>
</tr>
<tr>
<td>run 0 0 1 0.5 0</td>
</tr>
<tr>
<td>move 0 0 0.5 1 0</td>
</tr>
<tr>
<td>fast 0 0 0 0 1</td>
</tr>
</tbody>
</table>
Skip-gram model to generate the word similarity matrix. In learning process, the initial parameter “window size” is set to five (window\_size=5), which means that for each target word, the calculation sees in both directions, front and back, the five words neighboring the target from their contained sentence. The dimension of the feature vectors is set to be 100 (vector\_size=100). Since considering all words appearing in the document pool requires ample computation, the scope narrows to the words that appear at least four times (min\_count=4). After the learning, the word similarity matrix is obtained. Figure 4 shows the similarity value between two Korean words, “railroad” and “subway” and some words of high similarity scores with “railroad”.

The centrality score of a query document necessitates a network representation among the words contained in the query document. As mentioned before, the representation network is constructed by the PMI score. That is, any pair of nodes corresponding words in the document are linked with an edge if the score of (1) for the pair of corresponding words is over a given threshold. Under the constructed network, the PageRank scores are computed for all nodes by (2), and forms an entry of the centrality vector for the document. This research uses an open source network analysis and visualization tool, Gephi\(^4\) [1] because it has the PageRank module for the computation of centrality. The software generates centrality score for all words, but many low scores are not much meaningful because there exist only the small number of central nodes with high scores in a word network so that their words may be considered to be important. Since we use the centrality score to identify important

\(^{4}\)https://gephi.org
words in a document and therefore use them representatives of that document, the half lower scores are replaced with a value zero for simplicity.

This paper proposes the extended centrality score of (4)-(5) as a novel measure for a document representation. To prove its effectiveness, it is compared with two old types of representation methods; the methods using the occurrence and the centrality vectors. The former is just a Boolean vector representing word occurrences in the query document, while the latter the PageRank centrality vector. For each method of the three, the query vector, represented with that method over the query document, generates the document similarity scores together with the fd-idf vectors for all the documents in the pool. Those scores describe the extent to which the query documents is close to the documents in the pool. A document search engine may use them to display the documents related to the the query document. For each method, this experiment searches and arranges the documents in the pool in the descending order of the similarity scores, and eventually produces a recommendation list of documents as its search result. Here we considers all relevant documents as a list and consider their ranks in measurement.

Three measures are applied to the three search recommendation lists in order to compare the performance of our proposed method with those of the other two: the rank sum, the weighted rank sum, and the average precision. The rank sum is the sum of all ranks of the documents in a recommendation list having a relationship with the query document. It is formally denoted by \( \sum_{i \in L} w_i r_i \) where \( L \) is the set of documents in the recommendation list, \( r_i \) the rank of the document \( i \) in \( L \), and \( w_i \) is a weight whose value is 1 if the document \( i \) has any relationship with the query document and otherwise 0. The weighted rank sum has mathematically the same definition as the rank sum with different weights. Its weights are defined as values 1, 0.5 and 0; 1 if \( i \) has a strong relationship with the query document; 0.5 if \( i \) as a weak relationship with the query document; 0 for no relationship. The “average precision” is the 11-point interpolated average precision \(^5\) scores.[15] The performance of information retrieval is usually represented by the concepts of precision and recall, and their relationship is depicted by the precision-recall curve. The appropriate measure to evaluate performance of information retrieval can be obtained by how to describe a precision-recall curve as a representative number. Since the curve is a saw-tooth-shaped decreasing function, if we add one more document in a list, it increases the recall score but may decrease the precision score value depending on whether the added is relevant or not. To mitigate this unexpected jiggles, the interpolated precision is defined as:

\[
p_{int}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})
\]

Then eleven points for recall value, between zero and one, are selected such as 0.0, 0.1, \ldots, 1.0, and the interpolated precisions at those recall points calculated. The arithmetical average of the eleven interpolated precision values becomes the 11-point interpolated average precision score; that is, formally \( \sum_{i=1}^{11} p_{int}(r_i)/N \), where \( N = 11 \) with eleven recall values \( r_1, r_2, \ldots, r_{11} \). Table 2 shows the resulted measure scores for a query document (rows of the table), BTS , with the three methods (columns of the table). From the numbers in the table, the recommendation lists with the Word2vec centrality have the least values in the measures of the rank sum and the weight rank sum, and the highest values in the measure of average precision.

Documents having any relationship with the query document would be counted in the rank sum. If a document \( i \) is ranked high in the recommendation, i.e. its rank value \( r_i \) is low since the first rank is the highest, then the rank sum score becomes low. If the recommendation list however contains fewer documents with relationship, then it also lower the rank sum score because its formula contains fewer non-zero values. Lower scores in the rank sum, therefore, imply that either fewer related documents are captured by the recommendation list or the captured documents are relatively ranked higher. With the similar argument, low scores in the weighted rank sum might be interpreted as either more higher ranked strongly related documents or the fewness of related documents. In this respect, the two measures are not complete. When they are combined with the 11-point interpolated average precision, the performance is correctly interpreted. The precision is the ratio of the number of related documents to the size of the recommendation list. In our case, the 11-point interpolated average precision is used in order to give more weights to highly ranked related documents. If its score is higher, a method is interpreted to select more related documents and has more chance to have related documents in high rank positions. Thus when the scores of the rank sum and the weighted rank sum are lower and that of the average precision is higher, the method is likely to select more related documents which are better chance to be located in higher rank positions.

The results in Table 2 therefore say that the proposed method of the Word2vec centrality finds more documents having relationship with the query documents, and the found documents are more probable to locate in higher rank positions; the rank sum and the weighted rank sum scores are

![Figure 5: Rank sum (blue) and weighted rank sum (red) measure scores for BTS (The smaller is better)](image)

![Figure 6: 11-point average precision scores for BTS (The greater is better)](image)

lower and the average precision scores are higher in the two query cases. From this fact, it is concluded that the performance of the proposed method is better than the compared methods, the methods using the simple occurrence vectors and the centrality vectors. Also, documents having strong relationship contribute more heavily to the weighted rank sum scores. The rank scores for the proposed method are the lowest, and the method reflects the strength relatively better.

5. CONCLUSION

While conventional research on document search has mainly been keyword-based, this paper focuses on document-based search, search using a query of a document instead of a query of keywords. In the document-based search, the critical issue must be how to compare a given query document with documents in a pool. The conventional vector space model has some limitations in the comparison of documents because it is basically a scheme comparing keywords even though a document can be considered to be a sequence of words. A most important missing ingredient is that this approach ignores some context information expressed in a document, which cannot be explained with a collection of words. The main contribution of this paper is the proposal of a novel scheme to search documents using query documents. That is, though documents in a pool is represented as tf-idf vectors as in the conventional approach, their information is absorbed by the neural language model, and contained in a construct, the similarity matrix, together with the centrality information of the query document in the process of representation. The second contribution is that this scheme reflects the contextual information of documents. In the learning process of the neural language model, indirectly related but contextually and semantically close words are considered in the calculation of similarity, and that information is reflected on the calculation of the extended centrality vector for query documents, to be used later in the comparison with documents. By the experiment, this proposed scheme is proved to perform well in document search and to improve the search experience.

Still the proposed method has some limitations. While it reflects contextual information in search, it cannot consider the information of occurrence order of words because different occurrence orders can reflect different semantic importance. Also, the used neural language model has relatively small scope in capture of contextual information. When documents of big size are used as queries in search, the method cannot still overcome the degradation of similarity scores which might drop the performance of search. These can be issues for future research.

6. ACKNOWLEDGMENTS

This research was supported and funded by Korea Railroad Research institute.

7. REFERENCES
