Query Expansion Using Augmented Terms in an Extended Boolean Model

Tuan-Quang Nguyen, Jun-Seok Heo, Jung-Hoon Lee,
Yi-Reun Kim, and Kyu-Young Whang
Department of Computer Science
Korea Advanced Institute of Science and Technology (KAIST)
{quangntta, jsheo, handol, yrkim, kywhang}@mozart.kaist.ac.kr

We propose a new query expansion method in the extended Boolean model that improves precision without degrading recall. For improving precision, our method promotes the ranks of documents having more query terms since users typically prefer such documents. The proposed method consists of the following three steps: (1) expanding the query by adding new terms related to each term of the query, (2) further expanding the query by adding *augmented terms*, which are conjunctions of the terms, (3) assigning a weight on each term so that augmented terms have higher weights than the other terms. We conduct extensive experiments to show the effectiveness of the proposed method. The experimental results show that the proposed method improves precision by up to 102% for the TREC-6 data compared with the existing query expansion method using a thesaurus proposed by Kwon et al. [Kwon et al. 1994].

Categories and Subject Descriptors: Software and Applications

General Terms: Information Retrieval

Additional Key Words and Phrases: Extended Boolean Model, Query Expansion, Term Reweighting

1. INTRODUCTION

Search engines have become the main means for retrieving information on the Internet. Search engines receive a combination of terms (i.e., words) as the query from the user, and then, return documents relevant to the query as the result [Baeza-Yates and Berthier 1999]. The effectiveness of the search engines is mainly evaluated by precision and recall. Precision measures the ability to retrieve relevant documents among the returned documents. Recall measures the ability to retrieve relevant documents among all the relevant documents [Baeza-Yates and Berthier 1999]. Search engines are better if precision and recall are higher (i.e., the search engines retrieve more relevant documents in the result) [Baeza-Yates and Berthier 1999].

It is very difficult to make a query to completely represent the user's intension since the user's vocabulary is different from the Information Retrieval (IR) system's vocabulary [Xu and Croft 2000]. The terms used in the query may not match those used in the documents that are stored in the search engines (it is called "the mismatch problem" [Xu and Croft 2000]). For example, suppose the user wants to retrieve documents related to car. The user's query contains one term "car." Documents containing the term "car" and/or the term "automobile" are relevant to car. However, the search engine returns only those documents containing the term in the query (i.e., "car"). Thus, the retrieved documents do not satisfy the user's intension. This mismatch problem generally reduces the precision and recall of the search engines [Xu and Croft 2000].

Query expansion is a solution for solving the mismatch problem [Xu and Croft 2000; Kwon et al. 1994]. The key idea of query expansion is to generate a new query called the *expanded query* that contains not only the terms of the user's query but also the ones relevant to the query [Xu and Croft 2000; Kwon et al. 1994; Mandala et al. 1999]. To achieve this goal, the query is expanded by adding new terms that are relevant to the query [Xu and Croft 2000; Kwon et al. 1994; Mandala et al. 1999]. Query expansion generally increases the recall of the search engines because it allows the search engines to retrieve more relevant documents from the document collection [Kwon et al. 1994]. But it may decrease the precision because it is difficult to find relevant documents among retrieved documents [Kwon et al. 1994].

There have been a number of studies on query expansion [Baeza-Yates and Berthier 1999; Kwon et al. 1994; Salton et al. 1984]. The query expansion using Domain Adapted WeIghted Thesaurus (DAWIT) [Kwon et al. 1994] expands the user's query by using terms in the thesaurus. The query expansion using relevance feedback [Baeza-Yates and Berthier 1999; Salton et al. 1984] expands the user's query by using terms in the most recently retrieved documents. These methods generally enhance recall, but reduce precision [Kwon et al. 1994].

In this paper, we propose a query expansion method called the *query expansion using augmented terms* that improves precision without degrading recall. This method utilizes the user's preference [Clarke et al. 2000] for improving precision. The user's preference indicates that users typically expect documents having more terms in a query are ranked high [Clarke et al. 2000]. Thus, the co-occurrence of the query terms in documents plays a major role in ranking documents. By using the user's preference in query expansion, we are able to enhance the precision. The proposed method expands the query by adding *augmented terms* to the query. The augmented terms are conjunctions of the query terms for expressing their co-occurrence. We also suggest a term reweighting scheme called the *co-occurrence aware term reweighting scheme* that promotes ranks of documents having more augmented terms. This scheme assigns weights on the query terms so that augmented terms have higher weights than other terms.

The contributions of this paper are as follows: (1) we propose a query expansion method using augmented terms for the extended Boolean model, (2) we suggest a co-occurrence aware term reweighting scheme for the proposed query expansion method, (3) through extensive analysis and experiments, we show that the precision of our query expansion method is superior to those of existing query expansion methods, and in addition, the recall is as good as those of the existing ones.

The rest of this paper is organized as follows. Section 2 describes the extended Boolean model and prior work related to query expansion. Section 3 presents the motivation of this paper. Section 4 proposes the query expansion method using augmented terms. Section 5 presents the results of performance evaluation. Section 6 summarizes and concludes the paper.

2. BACKGROUND

In this section, we present the extended Boolean model [Salton et al. 1983] and describe existing query expansion methods in the model.

2.1 Extended Boolean Model (EBM)

The extended Boolean model combines the retrieval model of the Boolean model and the ranking model of the vector space model [Kwon et al. 1994; Salton et al. 1983].

In the Boolean model [Baeza-Yates and Berthier 1999], documents are represented as the sets of terms. Queries consist of the terms connected by three operators: AND, OR, NOT. For a given query, the model retrieves documents that satisfy Boolean expression of the query [Baeza-Yates and Berthier 1999].

In the vector space model [Salton and Lesk 1968], documents and queries are represented as vectors in a multi-dimensional vector space. The terms of the model form the multi-dimensional vector space. Each term in a document and a query is given a weight. Weights of terms are commonly calculated by "TF-IDF term weighting scheme" [Baeza-Yates and Berthier 1999]. In the TF-IDF term weighting scheme, a term has more weight if it frequently occurs in one document (i.e., having a high term frequency) and rarely appears in the rest of the document collection (i.e., having a low inverse term frequency) [Baeza-Yates and Berthier 1999]. Documents are ranked according to similarity of the documents to the query. Similarity is calculated by "cosine similarity measure" [Baeza-Yates and Berthier 1999], which is the cosine of angle between two vectors [Baeza-Yates and Berthier 1999]. The cosine similarity of a document $\vec{d}$ to a query $\vec{q}$ is calculated as in Eq. (1). In Eq. (1), the denominator is used to normalize the cosine similarity. Thus, the cosine similarity is, in fact, the inner product of the two vectors $\vec{d}$ and $\vec{q}$. The weights of terms in the query are initially set to 1.0 so that the similarity is the sum of the weights of the query terms in the document [Baeza-Yates and Berthier 1999].

$$sim(\vec{d},\vec{q}) = \frac{\vec{d} \bullet \vec{q}}{|\vec{d}| \cdot |\vec{q}|} \tag{1}$$

The extended Boolean model lies somewhat in between the Boolean model and the vector space model [Salton et al. 1983]. The extended Boolean model supports the Boolean query and document ranking. Figure 1 shows a retrieval model based on the

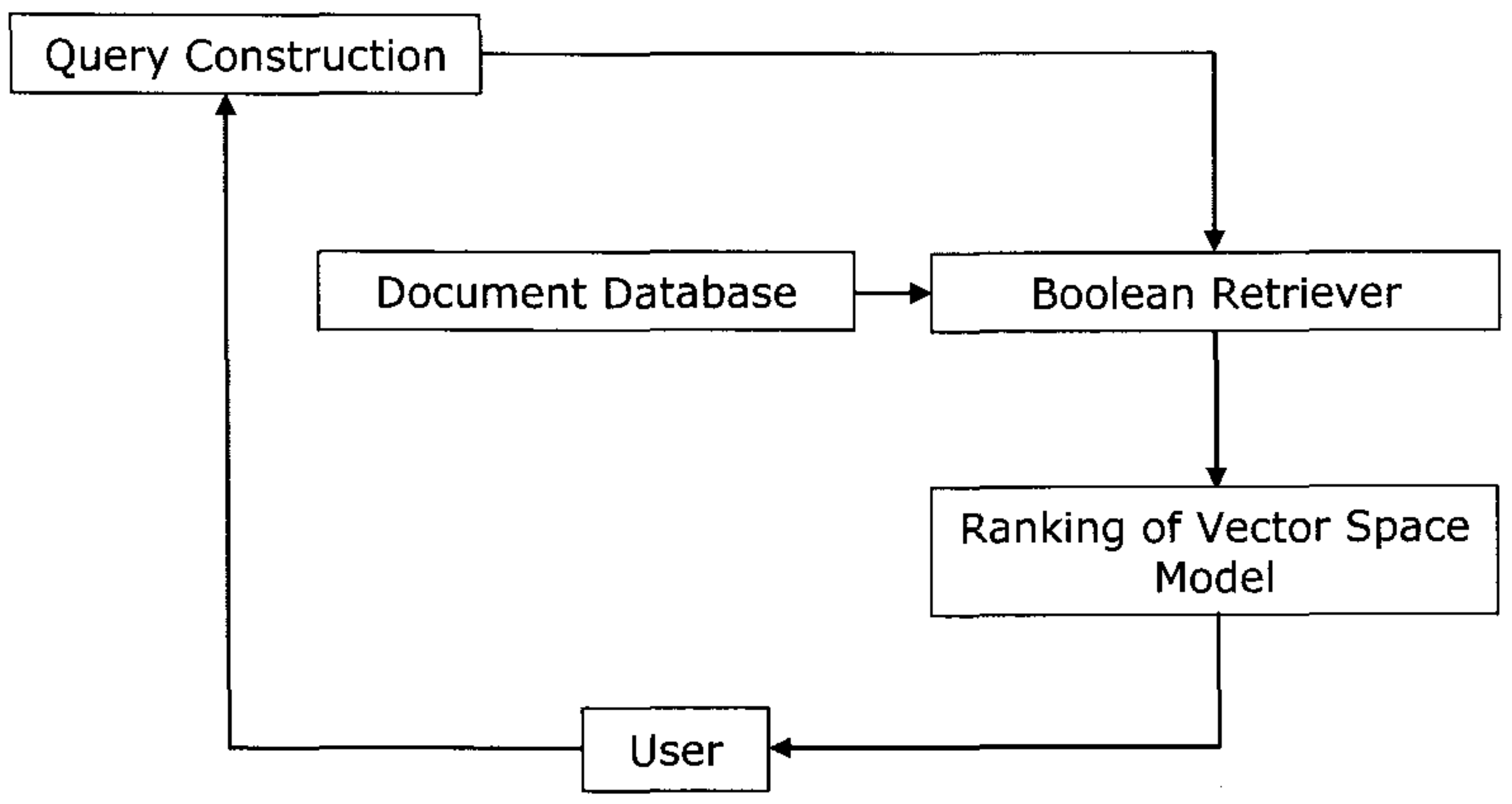


Figure 1. The retrieval model based on the extended Boolean retrieval model.

extended Boolean model. In Figure 1, the model combines the retrieval model of the Boolean model with the ranking model of the vector space model [Kwon et al. 1994; Salton et al. 1983]. All documents that satisfy the Boolean query are retrieved. Those documents are ranked by the cosine similarity measure.

In the extended Boolean model, to calculate the similarity of a document to a query, which is a combination of AND/OR operators, we process the query by recursively grouping the operators and terms into groups [Baeza-Yates and Berthier 1999]. Each group has its own order. Then, the similarity of the document to the query is calculated recursively according to the order of those groups [Baeza-Yates and Berthier 1999]. Suppose that $w_{A,q}$ and $w_{B,q}$ are the weights of terms A and B in the query, respectively. Suppose that $w_{A,d}$ and $w_{B,d}$ are the weights of terms A and B in the document, respectively. The similarity of the document to the query is calculated as in Eq. (2) [Baeza-Yates and Berthier 1999] for the two base cases (i.e., for AND and OR operators). The similarity depends on the weights of terms in the document and in the query. For a given IR system, the weights of the terms in documents are fixed. Thus, if we change the weights of terms in queries, the ranks of documents are also changed. In particular, if the weights of the terms in a query are increased, the similarity of documents that contain those terms to the query is also increased, and consequently, the ranks of those documents increase.

$$similarity(d, A_{w_{A,q}}\ AND\ B_{w_{B,q}}) = similarity(d, A_{w_{A,q}}\ OR\ B_{w_{B,q}})$$
$$= \frac{w_{A,q} \cdot w_{A,d} + \cdot w_{B,q} \cdot w_{B,d}}{2} \qquad (2)$$

EXAMPLE 1 Table I shows the information on a document collection. The document collection in this example contains two documents d_1 and d_2; d_1 contains two terms "petrol" and "car" d_2 contains one term "petrol". In the document d_1, the weights of the term "petrol" and "car" are 0.4 and 0.3, respectively. In the document d_2, the weight of the term "petrol" is 0.9. Let us consider two queries q_{or} = "car" or "petrol", q_{and} = "car" and "petrol". Suppose that the weight of "petrol" in q_{or} and q_{and} is 0.7 and

Table I. An example document collection.

Document (d)	Term	
	petrol	car
d_1	0.4	0.3
d_2	0.9	0.0

the weight of "car" in q_{or} and q_{and} is 0.8. In case of q_{or}, we retrieve d_1 and d_2 because those documents satisfy the Boolean expression of the query q_{or}. In case of q_{and}, we retrieve only d_1. Using Eq. (2), the similarities are calculated as in Eq. (3) and Eq. (4). Because *similarity*(d_2, q_{or}) is greater than *similarity*(d_1, q_{or}), the document d_2 is ranked higher than the document d_1 in case of q_{or}.

$$similarity(d_1, q_{and}) = similarity(d_1, q_{or}) = \frac{0.7\times0.4+0.8\times0.3}{2} = 0.26 \tag{3}$$

$$similarity(d_2, q_{or}) = \frac{0.7\times0.9+0.8\times0.0}{2} = 0.315 \tag{4}$$

2.2 Related Work

Query expansion improves the effectiveness of search engines by expanding the query using terms related to each term in the query [Kwon et al. 1994]. There have been a number of studies on query expansion. In this section, we explain them in more detail.

Kwon et al. [Kwon et al. 1994] proposed a thesaurus reconstructing method, called Domain Adapted Weighted Thesaurus (DAWIT), for enriching domain dependent terms in a thesaurus and proposed a simple query expansion using the thesaurus. The query expansion expands the query by adding new terms, called *related terms*, that are related to each term of the query. The authors used a thesaurus for finding related terms. The query expansion expands the query as in the following three steps. First, it finds related terms of each term in the query. Next, it replaces each term in the query with the disjunctions of the term and its related terms. Finally, it assigns a new weight to each term of the expanded query. This method uses the terms in the query and the related terms for expanding the query. However, it does not consider the user's preference.

Salton et al. [Salton et al. 1984] proposed the query expansion using relevance feedback. The query expansion using relevance feedback selects terms from the recently retrieved documents for query expansion. It combines the terms using AND and OR operators. This method uses AND operators to expand the query. But, it does not guarantee that documents having more query terms are ranked higher than other documents. Besides, it does not use the terms in the query to expand the query.

3. MOTIVATION

The query expansion generally reduces the precision of search engines [Kwon et al. 1994]. For a query that consists of disjunctions of terms, the query expansion in the extended Boolean model does not consider the user's preference. Here, the user's preference indicates that users prefer documents having more query terms [Clarke et

Table II. An example of query expansion that does not reflect the user's preference.

Document (d)	Term				Similarity (d, q_{exp})
	petrol	gas	car	automobile	
d_1	0.1	0.0	0.1	0.0	0.2
d_2	0.3	0.0	0.0	0.2	0.5
d_3	0.6	0.0	0.0	0.0	0.6

al. 2000]. In this paper, we exploit the user's preference on query expansion to improve precision without degrading recall of the search engines.

EXAMPLE 3.1 Suppose that there are three documents d_1, d_2, and d_3. d_1 contains "petrol" and "car"; d_2 contains "petrol" and "automobile"; d_3 contains "petrol." Let us consider the query q = "petrol" or "car". The term "gas" is related to the term "petrol." The term "automobile" is related to the term "car." In DAWIT, the expanded query is q_{exp} = ("petrol" or "gas") or ("car" or "automobile"). The weight of each term in the documents is shown in Table II, and the weight of each term in the query is 1.0. Table II shows the similarity of the documents to the expanded query. The document d_3 is ranked higher than the documents d_1 and d_2. However, according to the user's preference, the documents d_1 and d_2 should be ranked higher than the document d_3 because d_1 and d_2 contains two query terms while d_3 contains only one query term.

4. QUERY EXPANSION USING AUGMENTED TERMS

In this section, we present a new query expansion method, called query expansion using augmented terms, that reflects the user's preference.

The proposed method consists of the following three steps: (1) expanding the query by adding related terms; (2) further expanding the query by adding augmented terms; (3) assigning a weight to each term of the expanded query. In Section 4.1, we define the query model. From Section 4.2 to Section 4.4, we explain each step of the proposed method.

4.1 Query Model

In this paper, we deal with the query q that is a disjunction of m terms (t_1, t_2, ...,t_m) as in Eq. (5). Here, each term is a singleton. We define the term t_i ($1 \le i \le m$) as the *original term* and the query q as the *original query*. Table III defines the notation and terminology used in this paper.

$$q = t_1 \vee t_2 \vee \ldots \vee t_m \tag{5}$$

4.2 Expansion Using Related Terms

We present a method of expanding the query using related terms. We expand the query in two steps: (1) selecting related terms of the query from the thesaurus and (2) adding related terms to the query.

Table III. Notation and terminology used in this paper.

Symbols	Description
q	the user's query (or the original query)
Expanded Query(q)	the expanded query of the query q
Related Term(t)	the set of related terms of the term t
t_i	an original term in the query
t_{ij}	a related term of the original term t_i
τ	an augmented term
$w_{t,q}$	the weight of the term t in the query q

To select related terms, we first find candidate terms, and then, select related terms from the candidate terms. To find the candidate terms, we use the algorithm presented in the paper by Kwon et al. [Kwon et al. 1994]. Kwon et al. find candidate terms from an external thesaurus; all the terms that have relationships with terms in the query are candidate terms. We cannot select all candidate terms as related terms since excessively many related terms may reduce the effectiveness of search systems [Chung and Lee 2004; Nie and Jin 2002]. To select related terms, we propose two term selection strategies: *round robin* and *closest term.* Those strategies select related terms based on the similarity between terms. The similarity between terms are measured by the "mutual information" [Kwon et al. 1994]. In Section 4.4, we explain the "mutual information" in detail. Now, we describe the round-robin and closest-term strategies as follow:

- **Round-Robin Strategy:** we select related terms in the "round-robin order" [Silberschatz et al. 2003]. Let T_i ($1 \le i \le m$) be a set containing all candidate terms of the term t_i ($1 \le i \le m$) in the query. For each term t_i, we select a term in the set T_i with the highest similarity to the t_i. We conduct the above step repeatedly until we get the desired number of the related terms.
- **Closest-Term Strategy:** we select related term in the order of similarity. Let T be a set containing all candidate terms of the terms in the query. We select a term in T with the highest similarity repeatedly until we get the desired number of the related terms.

Now suppose that, for the term t_i ($1 \le i \le m$) in the query, p_i related terms are selected: t_{i1}, t_{i2}, ..., t_{ip_i}. The set of related terms of the original term t_i is described as in Eq. (6).

$$Related\ Term(t_i) = \{t_{i1}, t_{i2}, ..., t_{ip_i}\} \tag{6}$$

To add related terms to the query, we use the algorithm presented by Kwon et al. [Kwon et al. 1994]. We first use a disjunctive operator to combine terms in the query with their related terms. Thus, the term t_i is expanded to $t_i \vee t_{i1} \vee t_{i2} \vee \ldots \vee t_{ip_i}$. It is rewritten as $t_i \vee (\bigvee_{j=1}^{p_i} t_{ij})$. Consequently, we replace each original term in the query with disjunctions of the term and its related terms. The query in Eq. (5) is now expanded to a new query in Eq. (7).

$$Expanded\ Query_{Related}(q) = (t_1 \vee (\overset{p_1}{\underset{j=1}{\vee}} t_{1j})) \vee (t_2 \vee (\overset{p_2}{\underset{j=1}{\vee}} t_{2j})) \vee \ldots \vee (t_m \vee (\overset{p_m}{\underset{j=1}{\vee}} t_{mj})) \quad (7)$$

EXAMPLE 4.1 Let us consider the query q = "petrol" or "car." The terms "gas" and "oil" are candidate terms of "petrol." The terms "automobile" and "van" are candidate terms of "car." Suppose that the similarity between the pairs of ("gas", "petrol"), ("oil", "petrol"), ("automobile", "car"), and ("van", "car") are 0.9, 0.8, 0.7, and 0.6, respectively. In addition, suppose that we select two related terms for expanding the query. The round-robin strategy selects "gas" and "automobile" as the related terms because "gas" has the highest similarity to "petrol", and "automobile" has the highest similarity to "car." Thus, the expanded query becomes ("petrol" or "gas") or ("car" or "automobile"). The closest-term strategy selects "gas" and "oil" as the related terms because these terms have higher similarity than the other candidate terms. Thus, the expanded query becomes ("petrol" or "gas" or "oil") or ("car").

4.3 Expansion Using Augmented Terms

We have expanded the query by adding related terms. In this section, we first define the augmented term, and then, present the method of expanding the query using them. Suppose that we have the sets of related terms *Related Term*(t_i) as in Eq. (6) and the expanded query *Expanded Query Related*(q) as in Eq. (7).

We reflect the user's preference for expansion using augmented terms. Users prefer a document having $(n + 1)$ query terms to a document having n query terms [Clarke et al. 2000; Hiemstra, 1998]. Therefore, the number of query terms contained in the document is important. In other words, the co-occurrence of the query terms in the document is of major importance in ranking the results [Clarke et al. 2000]. We propose a new concept called the *augmented term* for expressing the co-occurrence of query terms. We use the *co-ordination level* [Clarke et al. 2000] as the number of query terms contained in the document. We define the augmented term in Definition 2 and the augmented-term co-ordination level in Definition 3.

DEFINITION 4.2 Let q be a query that is a disjunction of terms. Let R be a set of original terms and the related terms of the query q. Suppose that t is a term of the query q. A *query aspect* [Salton et al. 1983] t is the subset of R containing the term t and the related terms of t.

DEFINITION 4.3 Let q be a query that is a disjunction of terms. Let R be a set of the original terms and related terms of the query q. An *augmented term* τ is a conjunction of terms in R. Here, each singleton in τ belongs to one distinct *query aspect*.

DEFINITION 4.4 The *augmented-term co-ordination level* (*at-co-ordination level*) of τ is the number of singletons in τ.

EXAMPLE 4.5 Let us consider the query q = "petrol" or "car." The term "gas" is a related term of "petrol"; the term "automobile" is a related term of "car", R = {"petrol", "car", "gas", "automobile"}. There are two query aspects: the query aspect "petrol" is {"petrol", "gas"}, the query aspect "car" is {"car", "automobile"}. ("petrol" and "car"), ("petrol" and "automobile"), ("gas" and "car"), and ("gas" and "automobile") are augmented terms at at-co-ordination level 2 because they contain two singletons.

If "petrol" and "car" co-occur in a document d, we say that the document d contains the augmented term ("petrol" and "car").

Since augmented terms express co-occurrence of the query terms, we can use augmented terms to identify documents in which query terms co-occur. If a document contains an augmented term, the document also contains the singletons of the augmented term. For a given query q, the augmented terms of q is combined through the disjunctive operator because documents can satisfy only one or several augmented terms. Suppose that there are l augmented terms: $\tau_1, \tau_2, \ldots, \tau_l$. The disjunction of the augmented terms is $\tau_1 \vee \tau_2 \vee \ldots \vee \tau_l$. Now, we further expand the query in the previous section using the augmented terms. The final expanded query is described as in Eq. (8).

$$\begin{aligned} \textit{Expanded Query}(q) &= \textit{Expanded Query}_{Augmented}(\textit{Expanded Query}_{Related}(q)) \\ &= (t_1 \vee (\bigvee_{j=1}^{p_1} t_{1j})) \vee (t_2 \vee (\bigvee_{j=1}^{p_2} t_{2j})) \vee \ldots \vee (t_m \vee (\bigvee_{j=1}^{p_m} t_{mj})) \\ &\quad \vee (\tau_1 \vee \tau_2 \vee \ldots \vee \tau_l) \end{aligned} \tag{8}$$

Figure 2 shows the algorithm for expanding the query using augmented terms. We call this algorithm *Query Reformulation Using Augmented Terms.* In step 1, the algorithm uses either the round-robin strategy or the closest-term strategy to select related terms. The set of the related terms is T_R. Then, the algorithm combines the original terms of q with the related terms in T_R to create the set T_A of the augmented terms. The expanded query is initially set to be the query q. In step 2, the algorithm replaces each original term t in the query with the disjunction of t and its related

```
Algorithm Query Reformulation Using Augmented Terms:
Input:    The query $q$ that is a disjunction of terms
Output:   The expanded query $q_{exp}$
Algorithm:
Step 1. Initializing:
    1. 1 Get the set $T$ of the terms in $q$
    1. 2 Get the set $T_R$ of the related terms using either the round-robin or the closest-
         term strategy
    1. 3 Get the set $T_A$ of the augmented terms
    1. 4 Set the expanded query $q_{exp}$ to be $q$
Step 2. Expanding the query $q_{exp}$ using related terms: for each term $t \in T$
    2. 1 Replace $t$ with the disjunction of $t$ and its related terms in $T_R$
Step 3. Expanding the query $q_{exp}$ using the augmented terms: for each augmented term
        $\tau \in T_A$
    3. 1 Add $\tau$ to the expanded query $q_{exp}$ using the OR operator
```

Figure 2. The algorithm for expanding the query using augmented terms.

terms in T_R. In step 3, the algorithm adds augmented terms τ to the expanded query using the OR operator.

4.4 Co-occurrence Aware Term Reweighting

We now present a term reweighting scheme for our query expansion in such a way that important terms have higher weights. There are three types of terms in the expanded query: original terms, related terms, and augmented terms. For an original term, we set 1.0 as the weight. For a related term, we set the similarity between the original term and the related term as the weight. For an augmented term, we set the weight according to its at-co-ordination level and similarity.

For the weights of related terms, we use the algorithm presented by Kwon et al. [Kwon et al. 1994]. The weight of a related term is the similarity of the original term to the related term. Kwon et al. regard the "mutual information (MI)" as the similarity measure between terms. MI measures the information of x contained in y, and *vice versa.* The MI value used in Kwon et al. [Kwon et al. 1994] between two terms x and y is as in Eq. (10). The MI value is normalized by *log* (total number of terms in the document collection) in the range [0,1] [Chung and Lee 2004].

$$MI(x,\ y) = \log \frac{\dfrac{\#pair(x,y)\ in\ the\ document\ collection}{total\#}}{\left(\dfrac{\#x}{total\#}\right)\cdot\left(\dfrac{\#y}{total\#}\right)} \tag{9}$$

where *total#* is the total number of terms in the document collection

We present the method of assigning the weights to the augmented terms. According to the user's preference, the number of query terms (i.e., the co-ordination level) is of important role in the ranking results [Clarke et al. 2000]. Thus, the augmented terms at a higher at-co-ordination level are more important than those at a lower at-co-ordination level. We reweight the augmented terms in two steps: (1) setting a very high weight to an augmented term according to its at-co-ordination level; and (2) adding the sum of the weights of terms in the augmented term to its weight set in step (1) allowing differentiation of the weights of the augmented terms at the same at-co-ordination level.

Let us consider an augmented term τ. $|\tau|$ is its at-co-ordination level. To set the weight of an augmented term, we use a function that is monotonic to its at-co-ordination level. In addition, the weight of an augmented term at the at-co-ordination level $(n + 1)$ is always greater than that of an augmented term at the at-co-ordination level n [Clarke et al. 2000; Hiemstra 1998]. Thus, one of the function is $10^{|\tau|}$. For example, the function sets 100 to the weight of an augmented term at the at-co-ordination level 2 and 1000 to that of an augmented term at the at-co-ordination level 3.

Next, we use the similarity of terms in the augmented term τ to reweight the augmented terms. If each term in the augmented term is more important, the augmented term is more important. Thus, the weight of the augmented term depends on the sum of the weights of the terms in it. Consequently, the weight of the

```
Algorithm Co-occurrence Aware Term Reweighting:
Input:   The expanded query
Output:  The weights of terms in the expanded query
Algorithm:
Step 1. Initializing:
    1. 1 Get the set $T$ of terms in the expanded query
    1. 2 Classify terms in $T$ into three types: original terms, related terms, and
         augmented terms
Step 2. Reweighting terms: for each term $t$ in $T$
    2. 1 If $t$ is an original term, the weight of $t$ is 1.0
    2. 2 If $t$ is a related term, the weight of $t$ is computed as in Equation 9
    2. 3 If $t$ is an augmented term, the weight of $t$ is computed as in Equation 10
```

Figure 3. The algorithm for reweighting terms in the expanded query.

augmented term τ is calculated as in Eq. (10).

$$w_{\tau,q} = 10^{|\tau|} + \sum_{t \in \tau} w_{t,q} \tag{10}$$

Figure 3 shows the algorithm of reweighting terms in the expanded query. We call this algorithm *Co-occurrence Aware Term Reweighting.* In step 1, the algorithm extracts the set T of the terms of the expanded query and classifies them into three types: original terms, related terms, and augmented terms. In step 2, the algorithm calculates the weight of each term according to its type. If the term is an original term, its weight is 1.0. If the term is a related term, its weight is computed as in Eq. (10). If the term is an augmented term, its weight is computed as in Eq. (10).

EXAMPLE 4.6 Let us consider a query q.
q = "petrol" or "car" or "sale"
q_{exp} = *Expanded Query*(q)
= ("petrol" or "gas") or ("car" or "automobile") or
("sale" or "selling") or ("petrol" and "car") or
("petrol" and "automobile") or...or ("petrol" and "car" and "sale") or...
In step 1, we extract the set T of terms in the expanded query.
T = {"petrol", "car", "sale", "gas", "automobile", "selling",
("petrol" and "car"), ("petrol" and "automobile"),
("petrol" and "car" and "sale"), ...}

The original terms are "petrol", "car", and "sale." The related terms are "gas", "automobile", and "selling." The augmented terms are ("petrol" and "car"), ("petrol" and "automobile"), and ("petrol" and "car" and "sale")s, etc. In step 2, the algorithm computes the weight of each term in the expanded query q_{exp}. The weights of the terms "petrol", "car", and "sale" are 1.0 because they are original terms. The weights of the terms "gas", "automobile", and "selling" are computed to be 0.9, 0.8, and 0.7 as in Eq. (10). The weights of the augmented terms ("petrol" and "car"), ("petrol" and "automobile"), and ("petrol" and "car" and "sale") are calculated to be 100.95, 100.82, and 1001.1 as in Eq. (10). We can see that the weights of the augmented terms at the at-co-ordination level 3 (i.e., ("petrol" and "car" and "sale")) are higher than those of the augmented terms at the at-co-ordination level 2 (i.e., ("petrol" and "car") and ("petrol" and "automobile")). The weights of the augmented terms are higher than those of the original terms and the related terms. At the at-co-ordination level 2, the augmented term ("petrol" and "car") is weighted higher than the augmented term ("petrol" and "automobile").

5. PERFORMANCE EVALUATION

5.1 Experimental Data and Environment

The purpose of our experiments is to compare the effectiveness of the query expansion method proposed in this paper with one of the existing query expansion methods. We use *precision* and *recall* as the measures of the effectiveness. Precision is computed as in Eq. (11) [Baeza-Yates and Berthier 1999]. Recall is computed as in Eq. (12) [Baeza-Yates and Berthier 1999].

$$Precision = \frac{|\{Relevant\ documents\ in\ the\ result\}|}{|\{Documents\ in\ the\ result\}|} \tag{11}$$

$$Recall = \frac{|\{Relevant\ documents\ in\ the\ result\}|}{|\{Relevant\ documents\}|} \tag{12}$$

We compare precision and recall of the following five methods: QE using DAWIT (original), QE using DAWIT (round robin), QE using DAWIT (closest term), QE using augmented terms (round robin), and QE using augmented terms (closest term). Here, *QE using DAWIT* and *QE using augmented terms* are the query expansion methods using DAWIT [Kwon et al. 1994] and augmented terms, respectively. *Original, round robin*, and *closest term* are the strategies used for term selection. The original strategy is the one originally proposed by Kwon et al. [Kwon et al. 1994]. Here, all the terms related to the terms in the query are selected. The round-robin and closest-term strategies are those presented in Section 4.2. Nie et al. [Nie and Jin 2002] analyzed the impact of the number of the related terms in the expanded query on the effectiveness of search engines. Their experiments showed that the search engines are the most effective when the number of related terms in the expanded query ranges from 10 to 20. Thus, in our experiments, we use 15 related terms.

For the experimental data, we use the TREC-6 Adhoc Task English documents [Voorhees and Harman 1997], which is extensively used for the evaluation of information retrieval systems. The TREC data contain a total of 556,077 documents in the text format. These documents come from following sources: Financial Times (FT), Federal Register (FR), Congressional Record (CR), Foreign Broadcast Information Server (FBIS), LA Times (LAT). Table IV shows the statistics of the TREC-6 data.

In our experiments, we use 24 queries defined in the TREC-6 query collection

Table IV. Statistics of the TREC-6 data.

Contents	Size (MB)	Number (Docs)	Words/Doc (median)	Words/Doc (mean)
FT	564	210,158	316	412.7
FR	395	55,630	588	644.7
CR	235	27,922	288	1,373.5
FBIS	470	130,471	322	543.6
LAT	475	131,896	351	526.5

Table V. Statistics of the queries used in the experiments.

	Min Length (words)	Max length (words)	Mean length (words)
Query length	2	4	2.8

Table VI. Statistics of the Wordnet thesaurus.

Part of speech	Number of words
Noun	117,097
Verb	11,488
Adjective	22,141
Adverb	4,601
Total	155,327

[Baeza-Yates and Berthier 1999; Voorhees and Harman 1997]. The TREC-6 query collection consists of 50 queries. Among the 50 queries, 24 queries are generally used for the query expansion experiments [Voorhees and Harman 1997]. Thus, we also use those queries in our experiments. Table V shows the statistics of the query used in the experiments.

In query expansion, we use the WordNet thesaurus [Fellbaum 1998] for finding related terms. This thesaurus is an English thesaurus developed by Cognitive Science Laboratory of Princeton University. Table VI shows the statistics of the WordNet thesaurus.

To compare the effectiveness, we perform two experiments. The first experiment measures the average precision and recall of the queries while varying the number of retrieved documents from 10 to 100. We compute the average precision and recall in the following three steps: (1) getting the top 10, top 20, ..., top 100 retrieved documents in the results of each query, which belongs to the 24 queries used in the experiments; (2) computing precision and recall for each result in the previous step; and (3) computing the average precision and recall over the 24 queries when the number of retrieved documents is 10, 20, ..., 100. The second experiment measures the average precision while varying the average recall from 0% to 100% by controlling the number of retrieved documents. The result of this experiment is called "the precision/recall graph" [Baeza-Yates and Berthier 1999] which is a standard graph to compare the effectiveness of search engines. The results of the precision/recall graph are calculated in the following three steps: (1) for each query, measuring precision when recall = 10%; (2) computing the average precision over the 24 queries; and (3) conducting the steps (1) and (2) repeatedly while varying the recall from 20% to 100%.

We use the Odysseus ORDBMS [Whang et al. 2005] as search engines for all the experiments. The Odysseus ORDBMS uses the extended Boolean model mentioned in Section 2.1. The Odysseus ORDBMS calculates the similarity of the document to the query as shown in Eq. (13).

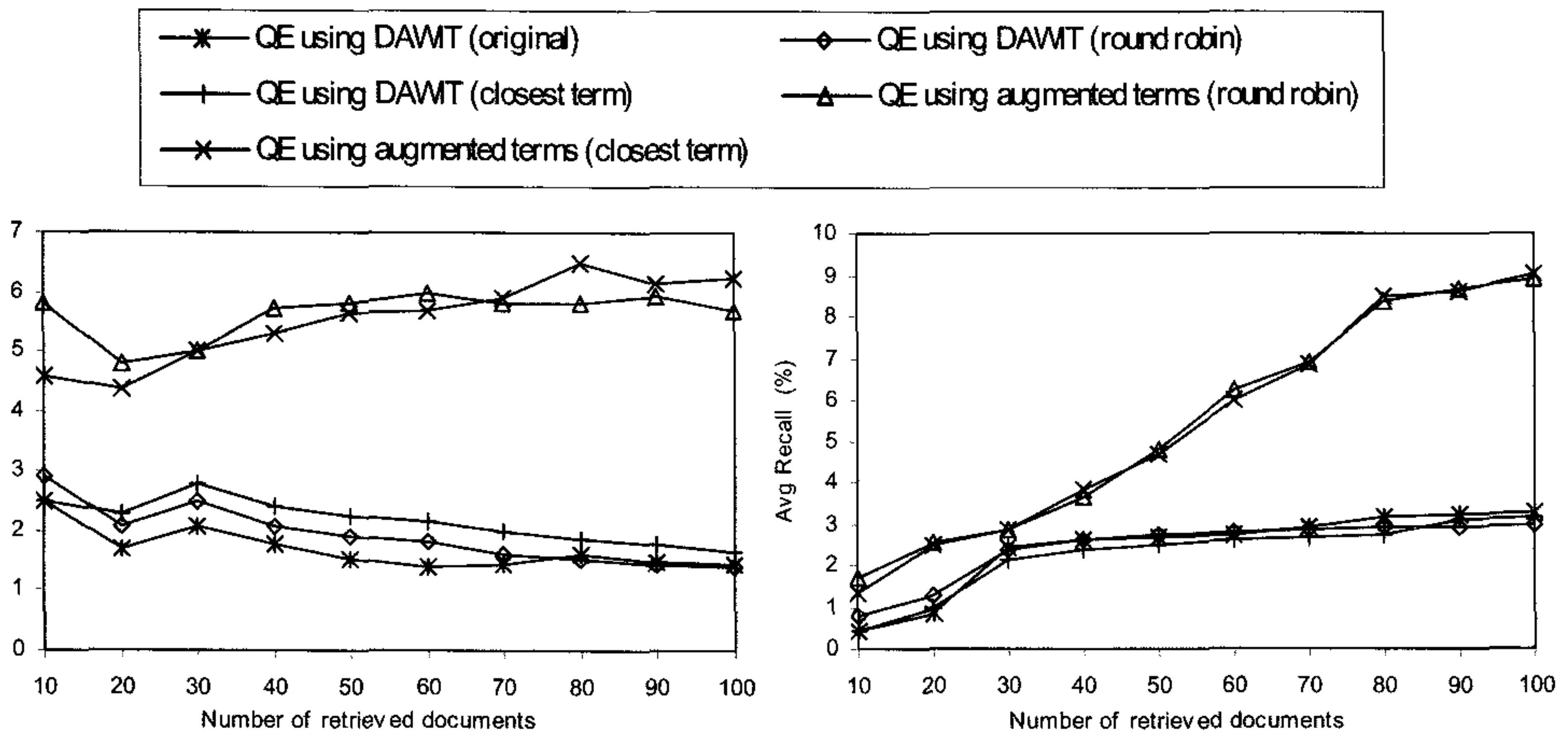
$$\begin{aligned} similarity(d, A_{w_{A,q}}\ AND\ B_{w_{B,q}}) &= similarity(d, A_{wA,q}\ OR\ B_{wB,q}) \\ &= w_{A,q} \cdot w_{A,d} + w_{B,q} \cdot w_{B,d} \end{aligned} \tag{13}$$

We conduct all the experiments on a Pentium 1.6 GHz Linux PC with 512 MBytes of main memory and 200 GBytes Segate E-IDE disks. We implement those query expansion methods in the C language and compile them by gcc version 3.2.2.

5.2 Results of the Experiments

This section presents the results of our experiments. The first experiment compares the average recall and precision values of the two query expansion methods. The second experiment compares the precision/recall graphs of the two query expansion methods.

Figure 4 shows precision and recall of the five query expansion methods as the number of retrieved documents is varied. In Figure 4(a), the original strategy achieves less precision than the round-robin and closest-term strategies do because the number of related terms used by the original strategy is larger than those used by the others. It was argued by Nie et al. [Nie and Jin 2002] that increasing the number of terms tends to reduce precision. In Figure 4(a), QE using augmented terms improves precision compared with QE using DAWIT regardless of the term selection



(a) Precision. (b) Recall.

Figure 4. Precision and recall as the number of retrieved documents is varied.

strategy. Figure 4(a), as mentioned in Section 1, shows that the precision of QE using the augmented terms increases, while that of QE using DAWIT decreases as the number of retrieved documents increases. When the number of retrieved document increases, both QE using DAWIT and augmented terms retrieve more relevant documents. For QE using DAWIT, the increase in the number of retrieved relevant documents is smaller than the increase in the number of retrieved documents. Thus, the precision of QE using DAWIT decreases as the number of retrieved document is varied. For QE using augmented terms, the increase in the number of retrieved relevant documents is greater than the increase in the number of retrieved documents. Thus, the precision of QE using augmented terms increases as the number of retrieved document increases. In Figure 4(b), as mentioned in Section 1, QE using augmented terms does not degrade recall compared with QE using DAWIT regardless of the term selection strategy. Figure 4(b) shows that the recall of QE using augmented terms is always superior to that of QE using DAWIT as the number of retrieved documents is varied from 10 to 100.

Figure 5 shows the precision of the five query expansion methods as recall is varied (i.e., "the precision/recall graph" [Baeza-Yates and Berthier 1999]). As mentioned in Section 1, Figure 5 shows that QE using augmented terms improves precision compared with QE using DAWIT regardless of the term selection strategy. Figure 5 shows that the precision of QE using augmented terms is superior to that of QE using DAWIT as recall is varied from 0% to 50% and is similar as recall is varied from 60% to 100%. In general, QE using augmented terms retrieves more relevant documents than QE using DAWIT because QE using augmented terms has a higher precision. When the recall is high, the precision is low because the number of the

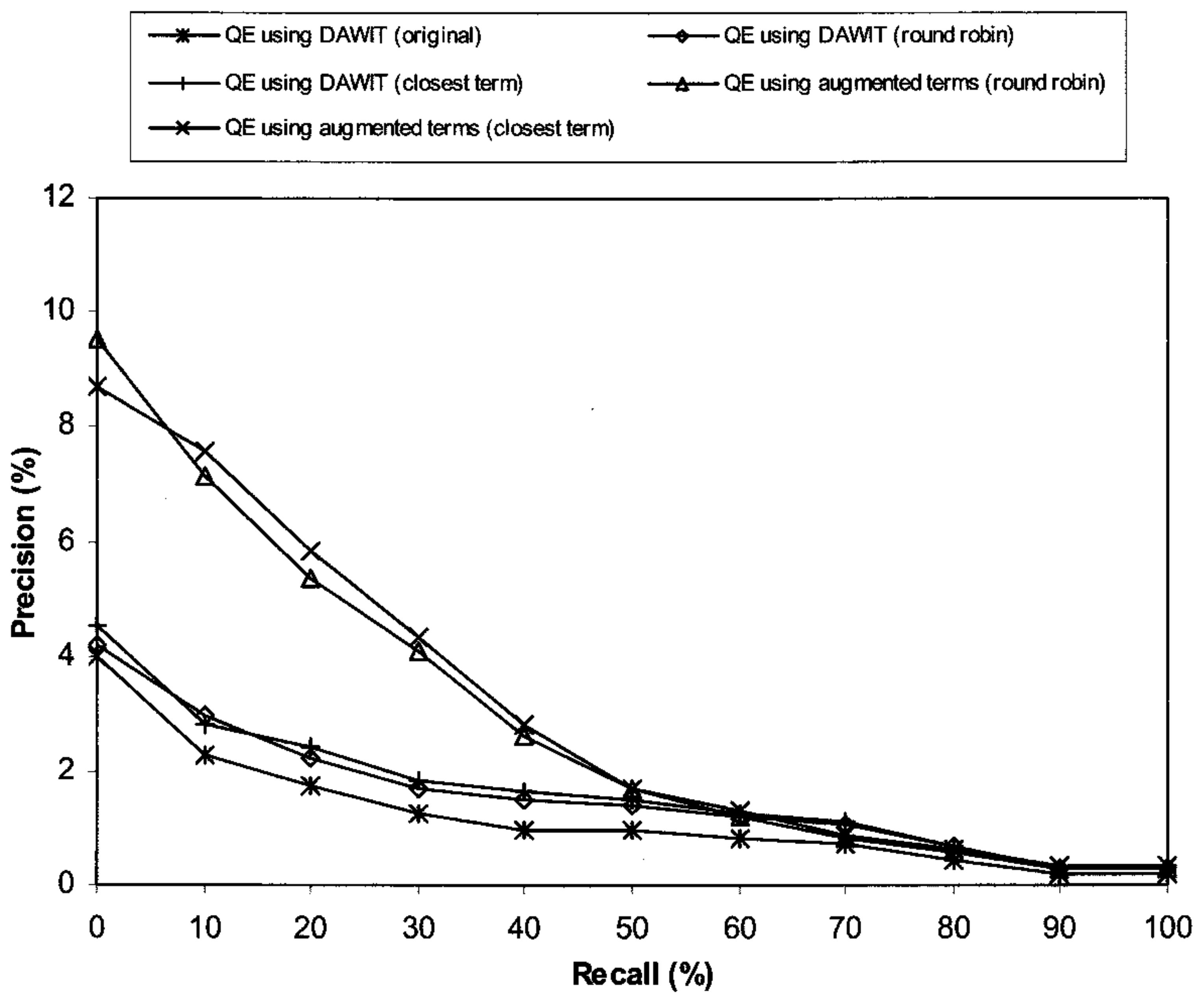


Figure 5. Precision as recall is varied (precision/recall graph).

retrieved documents is very large compared to that of relevant documents. It explains why QE using DAWIT and QE using augmented terms produce a similar precision when the recall is greater than 60%.

6. CONCLUSIONS

We have proposed a query expansion method using augmented terms in the extended Boolean model. We exploit the user's preference in query expansion. We first define the augmented terms for representing the user's preference in a query, and then, propose a method that expands a user's query by using the augmented terms. Thus, the proposed query expansion has the merits of both query expansion and the user's preference (i.e., enhancement of recall and precision, respectively).

We have proposed the co-occurrence aware term reweighting scheme based on the user's preference. The proposed scheme assigns a weight to each term of the expanded query in such a way that the augmented terms have higher weights than other terms. Consequently, the rank of a document gets higher as the document has more terms in the query.

Through extensive analysis and experiments, we have shown that the effectiveness (i.e., precision and recall) of our query expansion method is superior to that of an existing query expansion method. Experimental results using the TREC-6 document collection show that the query expansion using augmented terms outperforms the query expansion using DAWIT [Kwon et al. 1994] by up to 102% in precision and by up to 157% in recall for top-10 retrieved documents.

ACKNOWLEDGEMENTS

The authors are thankful for the incisive comments by Oh-Woog Kwon that helped make this paper more precise and readable. Tuan-Quang Nguyen was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2005-211-D00132). Jun-Seok Heo, Jung-Hoon Lee, Yi-Reun Kim, and Kyu-Young Whang were supported by the Korea Science and Engineering Foundation (KOSEF) through the National Research Lab Program funded by the Ministry of Science and Technology (MOST).

REFERENCES

BAEZA-YATES, R. AND RIBEIRO-NETO, B., *Modern Information Retrieval*, Addison Wesley, 1999.

XU, J. AND CROFT, W. B., "Improving the Effectiveness of Information Retrieval with Local Context Analysis," *ACM Transactions on Information Systems (TOIS)*, Vol. 18, No. 1, pp. 79–112, Jan. 2000.

KWON, O. W., KIM, M. C., AND CHOI, K. S., "Query Expansion Using Domain Adapted, Weighted Thesaurus in an Extended Boolean Model," In *Proc. 3rd Int'l Conf. on Information and Knowledge Management*, pp. 140-146, Gaithersburg, Maryland, Nov. 1994.

MANDALA, R., TOKUNAGA, T., AND TANAKA, H., "Combining Multiple Evidence from Different Types of Thesaurus for Query Expansion," In *Proc. 22nd Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 191–197, Berkeley, Aug. 1999.

SALTON, G. AND VOORHEES, E., "A Comparison of Two Methods for Boolean Query Relevancy Feedback," *Information Processing & Management*, Vol. 20, No. 5, pp. 637–651, Sept. 1984.

CLARKE, C. L. A., CORMACK, G. V., AND TUDHOPE, E. A., "Relevance Ranking for One to Three

Term Queries," *Information Processing & Management*, Vol. 36, No. 2, pp. 291–311, Mar. 2000.

SALTON, G., FOX, E. A., AND WU, H., "Extended Boolean Information Retrieval," *Communications of the ACM*, Vol. 26, No. 12, pp. 1022–1036, 1983.

SALTON, G. AND LESK, M. E., "Computer Evaluation of Indexing and Text Processing," *Journal of the ACM*, Vol. 15, No. 1, pp. 8–36, Jan. 1968.

CHUNG, Y. M. AND LEE, J. Y., "Optimization of Some Factors Affecting the Performance of Query Expansion," *Information Processing & Management*, Vol. 40, No. 6, pp. 891–917, Nov. 2004.

NIE, J. AND JIN, F., "Integrating Logical Operators in Query Expansion in Vector Space Model," In *Proc. ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, Tampere, Finland, Aug. 2002.

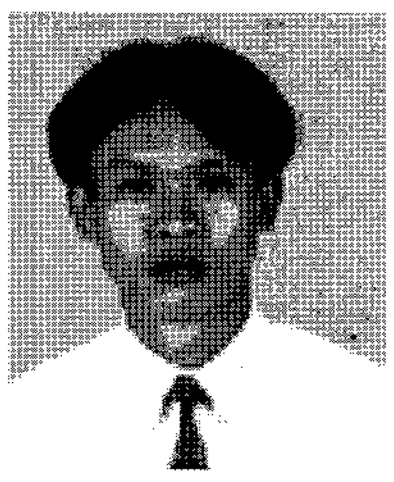
SILBERSCHATZ, A., GALVIN, P. B., AND GAGNE, G., *Operating System Concepts*, Wiley, 2003.

HIEMSTRA, D., A Linguistically Motivated Probabilistic Model of Information Retrieval, In *Proc. The 2nd European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pp. 569–584, Crete, Greece, Sept. 1998.

VOORHEES, E. M. AND HARMAN, D., "Overview of the Sixth Text Retrieval Conference (TREC-6)," In *Proc. The 6th Text REtrieval Conference*, pp. 1–24, Gaithersburg, Maryland, Nov. 19-21, 1997.

FELLBAUM, C., *WordNet – An Electronic Lexical Database*, MIT Press, 1998.

WHANG, K., LEE, M., KIM, M., AND HAN, W., "Odysseus: a High-Performance ORDBMS Tightly-Coupled with IR Features," In *Proc. IEEE 21th Int'l Conf. on Data Engineering (ICDE)*, pp. 1104–1005, Tokyo, Japan, Apr. 5–8, 2005.

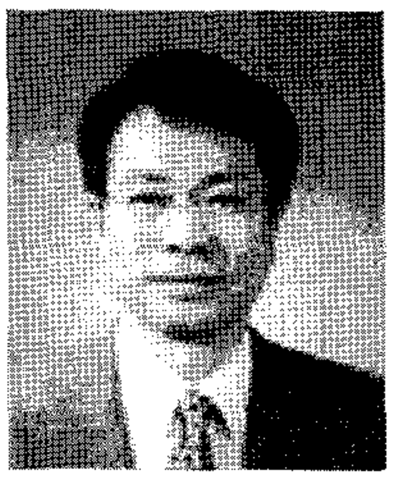


Tuan-Quang Nguyen received the B.S. degree in computer science from Hanoi University of Technology (HUT) in 2001 and the M.S. degree from Korea Advanced Institute of Science and Technology (KAIST) in 2007. From 2000 to 2005, he worked for several information technology companies where he developed search engines and image processing systems. He is currently a senior researcher at Korea Wisenut Co., Ltd. His research interests are information retrieval and text mining.

Jun-Seok Heo received the B.S. (1995) and M.S. (1997) degrees in computer science and statistics from University of Seoul. From 1997 to 2002, he was a senior researcher at Daewoo Telecom Co., Ltd. and Mercury Co., where he participated in developing switching systems. In 2002, he was an entrusting researcher at the Advanced Information Research Center (AITrc). He is currently a Ph.D. Candidate in the Department of Computer Science at Korea Advanced Institute of Science and Technology (KAIST). His research interests include information retrieval, geographic information systems, and telecommunication systems.

Jung-Hoon Lee received his B.S. and M.S. degrees in computer engineering from Kyungpook National University, in 1995 and 1997, respectively. From 1997 to 2002, he was a senior research engineer at Korea Information and Communication Co., Ltd. In 2002, he was an entrusting researcher at the Advanced Information Research Center (AITrc). He is currently doing his Ph.D. in Korea Advanced Institute of Science and Technology (KAIST). His research interests include peer-to-peer and sensor network.

Yi-Reun Kim received the B.S. and M.S. degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST), in 1999 and 2001, respectively. He is currently a Ph.D. Candidate in the Department of Computer Science at KAIST. His research interests include storage systems and embedded DBMSs.



Kyu-Young Whang graduated (Summa Cum Laude) from Seoul National University in 1973 and received the M.S. degrees from Korea Advanced Institute of Science and Technology (KAIST) in 1975, and Stanford University in 1982. He earned the Ph.D. degree from Stanford University in 1984. From 1983 to 1991, he was a Research Staff Member at the IBM T. J. Watson Research Center, Yorktown Heights, NY. In 1990, he joined KAIST, where he currently is a full professor at the Department of Computer Science and the Director of the Advanced Information Technology Research Center (AITrc). His research interests encompass database systems/storage systems, object-oriented databases, multimedia databases, geographic information systems (GIS), data mining/data warehouses, XML databases, and data streaming. He is an author of over 100 papers in refereed international journals and conference proceedings (and over 150 papers in domestic ones). He served as an IEEE Distinguished Visitor from 1989 to 1990, received the Best Paper Award from the 6th IEEE International Conference on Data Engineering (ICDE) in 1990, served the ICDE six times as a program co-chair and vice chair from 1989 to 2003, and served program committees of over 100 international conferences including VLDB and ACM SIGMOD. He was the program chair (Asia and Pacific Rim) for COOPIS'98, the program chair (Asia, Pacific, and Australia) for VLDB 2000, and a program co-chair of ICDE2006. He is the general chair of VLDB2006, PAKDD 2003, and DASFAA 2004. He twice received the External Honor Recognition from IBM. He is the Coordinating Editor-in-Chief of the VLDB Journal having served the editorial board as a founding member for thirteen years. He was an associate editor of the IEEE Data Engineering Bulletin from 1990 to 1993, an editor of Distributed and Parallel Databases Journal from 1991 to 1995, and an associative editor of IEEE TKDE from 2002-2006. He is on the editorial boards of the WWW Journal and Int'l J. of GIS. He was a trustee of the VLDB Endowment from 1998 to 2004 and currently is a steering committee chair of the DASFAA Conference and a steering committee member of the IEEE ICDE and PAKDD Conferences. He served the IEEE Computer Society Asia/Pacific Activities Group as the Korean representative from 1993 to 1997. He is a Fellow of the IEEE, a member of the ACM, and a member of IFIP WG 2.6.