Query Expansion

Per Ahlgren

The purpose of query expansion, the process of adding new terms to a query, is to improve the retrieval effectiveness. Query expansion can be divided into three types: manual, interactive and automatic. This article presents, and to some extent discusses, these three types of query expansion. The article also gives examples of ranking algorithms for query expansion.

Information retrieval (IR) deals with various methods of storing, structuring and retrieving documents. An *IR system* is a computer-based system for storage and retrieval of documents. Normally, the retrieval mechanism of an IR system is based on matching terms, i.e. terms that occur both in the document and in the query.[1] Then, users of an IR system have to express their information needs in the vocabularies of desired documents. This implies a difficulty for the user, especially when concerning large full text databases, as these contain different expressions for the same concept (Voorhees 1994, p. 61). Relevant documents, in which search concepts are expressed by other terms than the ones used by the user, could remain unretrieved. In this case, the recall of the retrieval (the fraction of the relevant documents which has been retrieved) will suffer.

The aim of this article is to present a number of (already existing) methods for *query expansion* (QE). QE could be defined as the process of adding new terms to a query. QE results in a new query and aims at improving retrieval effectiveness.

QE

QE could be divided into three different types. If the user chooses expansion terms, it is called *manual QE*; if the system suggests expansion terms to the user, it is called *interactive QE*. If the whole process is invisible for the user, *automatic QE* is at hand.

One important issue to consider when applying QE is from which source the expansion terms should be collected. Efthimiadis (1996, p. 124) discusses two different types of sources: sources based on search results and sources constituting knowledge structures (independent of the search process). The documents which have been retrieved in an earlier iteration of the search and which have been judged to be relevant constitute an example of a source based on search results. Sources in the form of knowledge structures could be either dependent or non-dependent on the document collection. One example of a collection-dependent knowledge structure is an automatically constructed thesaurus. One example of a collection-independent knowledge structure is an area-specific (manually constructed) thesaurus. The three types of QE and possible term sources are shown graphically in Figure 1.

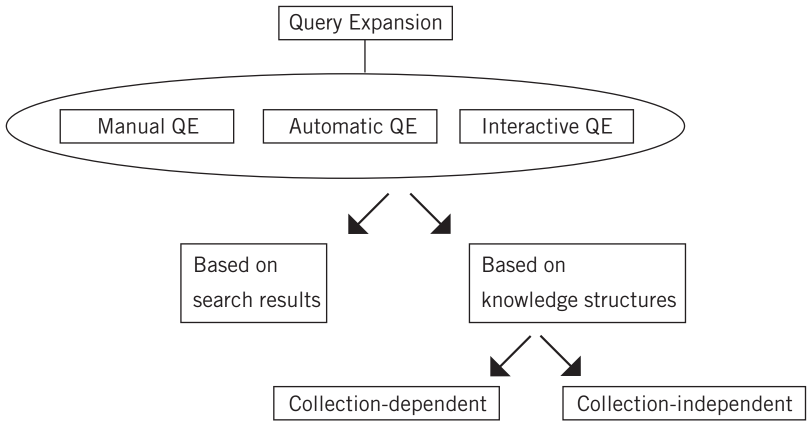

Figure 1. QE: types and term sources. Based on Efthimiadis 1996 (p. 124).

Relevance feedback is an iterative process in which the user judges the relevance of retrieved documents, after which the system uses this information in order to modify the query (Korfhage 1997, pp. 221–224). A typical relevance-feedback operation could be described as follows (Efthimiadis 1996, p. 134). The user constructs a query that is matched against the document collection. He then judges the relevance of retrieved documents, usually on the basis of titles or abstracts, and chooses a number of relevant ones. The system uses the chosen relevant documents in order to modify the initial query by re-weighting the initial query terms and/or by adding terms that appear useful and deleting terms that do not. This process creates a new query, which is more similar to the documents that were judged relevant than the initial query was.

There are a number of *ranking algorithms* for QE. These attempt to quantify the usefulness of a query term at retrieval (ibid., p. 139). Terms that are highly ranked by a certain algorithm could then be used in order to expand the query. Two of these ranking algorithms are described, and partially compared, below.

The *WPQ-algorithm*. The weight (the usefulness) of a term *t* is defined according to

$$WPQ(t) = \log \frac{(r+0.5)(N-n-R+r+0.5)}{(n-r+0.5)(R-r+0.5)} \cdot \left(\frac{r}{R} - \frac{n-r}{N-R} \right), \qquad (1)$$

where *N* is the number of documents in the collection, *R* the number of documents judged to be relevant by the user, *n* the number of documents indexed by *t* and *r* the number of documents judged by the user to be relevant and that have been indexed by *t*. The second factor in (1) puts relatively strong weight on the condition that terms occur in relevant documents. This is because the second component of the factor normally becomes so small that its information content is lost (Efthimiadis 1995, p. 608). The first factor – compared to the second one – shows greater consideration to the condition that terms occur infrequently in the documents that have not been judged to be relevant (the number of documents that have not been judged to be relevant and are indexed by a term *t* is equal to $n - r$).

Another example of a ranking algorithm is *Porter's algorithm*, defined according to

$$Porter(t) = \frac{r}{R} - \frac{n}{N} \quad (2)$$

In conformity with the second factor in (1), (2) puts a relatively strong weight on the condition that terms occur in relevant documents: the second component in (2) normally becomes so small that its information content is lost.

Example: Let t_1,t_2 and t_3 be terms and let q be a query. Let the collection consist of 1000 documents, i.e. N=1000 Assume that 10 documents are judged to be relevant with respect to q, i.e. R=10. Assume that the number of documents indexed by t_1 is 50 and that the number of documents that are judged to be relevant and that are indexed by t_1 is 8. Then

$$WPQ(t_1) = \log\frac{(8+0.5)(1000-50-10+8+0.5)}{(50-8+0.5)(10-8+0.5)}\cdot\left(\frac{8}{10}-\frac{50-8}{1000-10}\right)=1.42 \quad (3)$$

and

$$Porter(t_1) = \frac{8}{10} - \frac{50}{1000} = 0.75 \quad (4)$$

Assume that the number of documents indexed by t_2 is 10 and that the number of documents that are judged to be relevant and that are indexed by t_2 is 6. Then

$$WPQ(t_2) = \log\frac{(6+0.5)(1000-10-10+6+0.5)}{(10-6+0.5)(10-6+0.5)}\cdot\left(\frac{6}{10}-\frac{10-6}{1000-10}\right)=1.49 \quad (5)$$

and

$$Porter(t_2) = \frac{6}{10} - \frac{10}{1000} = 0.59 \quad (6)$$

Finally, assume that the number of documents indexed by t_3 is 10 and that the number of documents that are judged to be relevant and that are indexed by t_3 is 10. Thus, t_3 indexes all documents that have been judged relevant, but no other document. Then

$$WPQ(t_3) = \log \frac{(10+0.5)(1000-10-10+10+0.5)}{(10-10+0.5)(10-10+0.5)} \cdot \left(\frac{10}{10} - \frac{10-10}{1000-10} \right) = 4.62 \quad (7)$$

and

$$Porter(t_3) = \frac{10}{10} - \frac{10}{1000} = 0.99 \quad \textbf{(8)}$$

Both *WPQ* and *Porter* rank t_3 the highest, but the two algorithms differ with regard to the ranking of t_1 and t_2. *WPQ* ranks t_2 higher than t_1, while *Porter* ranks t_1 higher than t_2. *Porter* – as opposed to *WPQ* – puts almost all weight on the condition that terms occur in relevant documents. t_1 occurs in eight of the ten documents that are judged to be relevant, t_2 in six of them. This is reflected in *Porter's* ranking of the two terms. With respect to terms' absence from documents that have not been judged to be relevant, t_2 performs better than t_1, which *WPQ* takes into consideration.

Note: a term that is highly ranked by *WPQ* (or *Porter*) could be completely unsuitable as an expansion term. Assume that a term *t* does not occur in a query *q*, which is to be expanded. Assume, also, that *t* occurs in all documents that the user has judged to be relevant, but not in any other document in the collection (the term t_3 in the example above is such a term). Finally, assume that the retrieval strategy used is based on matching terms. Under these assumptions, both *WPQ* and *Porter* will rank *t* highly. To expand *q* with *t* would, however, be pointless, as relevant documents that have *not* already been retrieved do not match the expanded query with regard to *t*.

Efthimiadis (1996, p. 123) gives examples of research questions with reference to QE. Some of the issues raised are:

- Which are the best terms for QE?
- Where can we get the expansion terms?
- How useful could the terms be?
- How could we rank the terms?
- Are users capable of discovering good terms?

- How do users choose terms?
- What is the relation between the initial query terms and the terms that the users choose?
- Is there a difference between the terms chosen by the user and the ones presented by the system?

Manual QE

When searching in Boolean IR systems so-called *search strategies*, i.e. ways to tackle search problems, could be used (Harter 1986). One possibility for the user is to apply “the building blocks strategy”, a search strategy that could be described thus:

1. Identify the *n* concepts that should be present in a retrieved document.
2. For each concept (*n* concepts) identified in step 1, construct a list of synonymous (or nearly synonymous) terms, which express the concept.
3. For each list (*n* lists) constructed in step 2, combine the terms of the list with the operator OR.
4. Combine the *n* OR formulations constructed in step 3 with the operator AND.

In step 1, an initial term is generated for a given concept (the term that is used to identify the concept). In step 3, the initial term (which could be regarded as a query) is expanded with the synonymous (or nearly synonymous) terms that were generated in step 2 (and with the OR operator). This process is an instance of manual QE. The terms that are added to the initial terms in step 3 could be collected from a manually constructed (and collection-independent) thesaurus, for instance. In this case, manual QE based on a collection-independent knowledge structure is at hand.

Another possibility for the user is to apply the search strategy “citation pearl growing”. Step 1 in this strategy combines *n* terms with the operator AND, and the resulting AND formulation is matched against the document collection. Step 2 uses descriptors and free text terms from retrieved relevant documents ("the pearl"), together with the OR operator, in order to expand the initial terms (the conjuncts in the AND formulation in step 1). The resulting formulation is an AND formulation, the conjuncts of which are OR-formulations. The formulation is then matched against the document collection, and hopefully new relevant documents will be retrieved. (The process of examining retrieved relevant documents in order to produce new terms, which the conjuncts of the current AND formulation could be expanded with, could be iterated an arbitrary number of times.)

In step 2, the user expands an AND formulation with new terms, and thereby manual QE (based on search results) is at hand. Efthimiadis (1996, p. 128) points out that "citation pearl growing" could be regarded as the manual equivalent to relevance feedback.

Automatic QE

The four following sections will describe, in turn, (1) QE with regard to the vector model, (2) a method used to create association clusters around the query terms, (3) a method that ranks each term in the document collection that does not occur in the initial query with consideration to usefulness as expansion term, and (4) a method that uses a general lexical system.

QE and term re-weighting for the vector model

Relevance feedback was applied early on to the vector model. In this *IR model*[2] both a document d_j and a query q are represented by t-dimensional vectors:

$$d_j = \vec{d}_j = (w_{1,j}, w_{2,j}, \dots, w_{t,j}) \qquad (9)$$

$$q = \vec{q} = (w_{1,q}, w_{2,q}, \dots, w_{t,q}) \qquad (10)$$

where t is the number of terms in the *vocabulary*, i.e. the set of index terms in the system, $w_{i,j}$ the weight of the term k_i in the document d_j

and $w_{i,q}$ the weight of k_i in the query q.[3] The vector model measures the degree of similarity between query and document by measuring the similarity between the vectors in question. The *cosine measure* is often used for this purpose (see, for example, Baeza-Yates & Ribeiro-Neto 1999, pp. 27f.).

The basic idea concerning QE for the vector model is to modify the query vector $\vec{q}$ so that it becomes more similar to the vectors for the relevant documents. Two assumptions are made: (a) the vectors for documents that have been identified as relevant are similar, i.e. relevant documents are similar, and (b) the vectors for non-relevant documents have a weak similarity with the vectors for relevant documents.

Let D_r be the set of relevant documents (among the retrieved ones), D_n the set of non-relevant documents (among the retrieved ones), and let α, β and γ be constants. Further, let q be a query and $\vec{q}$ the corresponding vector. $\vec{q}$ is matched against the document vectors, and the documents are ranked on the basis of the generated similarity values. The user judges the relevance of, for example, the 20 most highly ranked documents. The set D_r will then contain the relevant documents among the 20, while the set D_n will contain the non-relevant ones among the 20. A modified query vector $\vec{q}_m$ could now be created, for example by means of the following equation (ibid., p. 119):

$$\vec{q}_m = \alpha\vec{q} + \frac{\beta}{|D_r|}\sum_{\forall d_j \in D_r}\vec{d}_j - \frac{\gamma}{|D_n|}\sum_{\forall d_j \in D_n}\vec{d}_j \qquad (11)$$

Let us assume that α, β and γ are equal to 1. The calculation of the modified query vector is made as follows. First, average vectors are generated. For each term k_i in the vocabulary, k_i's weights in the documents that have been judged relevant are summed. This is done by way of adding the vectors of these documents. The result of the addition is a new vector, where a given position contains the sum of the weights (in the relevant documents) of the corresponding term. Then

$$\frac{\beta}{|D_r|}$$

is multiplied with the new vector. (The denominator in the expression above is the number of objects in the set D_r, i.e. the number of documents judged to be relevant.) The result of this becomes a new vector. A given position in the vector contains the average weight of the corresponding term in the relevant documents. This new vector is the average vector for the relevant documents. Then the corresponding operations are executed with respect to the documents judged as non-relevant.

When the average vectors have been created, the difference between them is calculated. The result becomes a new vector, in which a given position contains the difference between the average weight of the corresponding term in the relevant documents and its average weight in the non-relevant ones. Finally, this new vector is added to the initial query vector. A given position in the resulting vector – the modified query vector $\vec{q}_m$ – contains the sum of the corresponding term's weight in the initial query and the difference between its average weight in the relevant documents and its average weight in the non-relevant ones.

The modified vector $\vec{q}_m$ is matched against the document vectors and the documents are ranked again on the basis of the generated similarity values. If desired, a new query vector could be created on the basis of relevance judgements with respect to the *new* ranking.

By using the initial query vector, initial query terms could be given a comparatively great importance in the modified query. If a term, that does not occur in q, occurs in one or more documents judged to be relevant, but not in any document judged to be non-relevant, the term will be added to q, i.e. its weight in the modified query vector becomes greater than 0. Further, the importance of a query term could be toned down in the new query. This could be done if the term occurs in few or no relevant documents but in a number of non-relevant ones. As a matter of fact, such a scenario could result in the weight of the term in the new query being 0 (or negative).[4] Such a term will not give any positive contribution to the similarity value for a document that contains the term (with regard to the new query).

The method described above is an example of QE based on search results.[5]

QE by local clustering

Baeza-Yates & Ribeiro-Neto (ibid., p. 125) describe a method that, like the method in the previous section, is based on search results, but that does not involve assistance from the user. The idea here is to use the documents retrieved by the initial query q to construct, for each query term (or word stem in the query), a *cluster* (i.e. a group) of terms (or word stems) which are correlated with the query term (the word stem). The correlation between terms could be defined in different ways. Once the clusters have been created, the query is expanded with the terms (or word stems) of the clusters.

The method works with *local association clusters*, and the following is a description of how these could be created. The example makes use of terms rather than word stems. Let D_l be the set of the documents retrieved by a query q. We will call the set in question the *local document set*. The *local vocabulary,* V_l, is defined as the set of all distinct words in D_l.

An association cluster is based on *co-occurrences* of terms in documents (two terms *co-occur* in a document if both terms occur in the document). The idea is that words that often co-occur in documents are semantically related. Let k_u and k_v be words in the local vocabulary V_l and let $f_{k_i,j}$ be the frequency of the term k_i in the document d_j. The following equation defines the correlation, $c_{u,v}$, between the two words with regard to the local document set D_l:

$$c_{u,v} = \sum_{d_j \in D_l} f_{k_u,j} \times f_{k_v,j} \qquad (12)$$

The correlation factor $c_{u,v}$ does not take into consideration whether the two words have similar frequencies in the documents or not. The factor is therefor said to be *non-normalised*. The following factor, however, takes this into consideration and is said to be *normalised*:

$$c_{u,v(norm)} = \frac{c_{u,v}}{c_{u,u} + c_{v,v} - c_{u,v}} \qquad (13)$$

If the two words have exactly the same frequencies in all documents considered, and if at least one frequency is higher than 0, we obtain the maximum correlation value, 1.

Now assume that we have measured the correlation for each pair of words in V_l according to one of the two alternatives above. We could place the correlation values in an *m* times *m association matrix* (where *m* is the number of words in V_l), i.e. a table where the words in the local vocabulary are represented both horizontally and vertically. A given cell in the matrix shows the correlation value for the words corresponding to the row and the column that the cell belongs to.

Given the matrix, we could create local association clusters according to the following: for each term k_u in V_l we start searching the row in the matrix that corresponds to k_u, i.e. row number *u*. Then we search for the *n* highest correlation values in this row, disregarding k_u's correlation with itself. Let $S_u(n)$ be the set of terms corresponding to the *n* highest correlation values. We find these terms by moving to the top of the column in which the value was found. We call $S_u(n)$ a *local association cluster around the term* k_u.

We can now expand the query *q*. For each term k_u in *q*, the terms (or some of them) of the local association cluster $S_u(n)$ are added to *q*. The new query, say q', will hopefully retrieve more relevant documents than the initial query *q*.

As we are interested only in obtaining association clusters around the query terms, the system does not have to perform the correlation calculations for two terms that do not occur in *q*. This means that the association clusters around the query terms could be generated quickly.

QE based on global analysis

The following will describe a method that involves an analysis of the *whole* document collection. The method is based on a collection-dependent knowledge structure and will be described rather informally (ibid., pp. 131–133).

The method involves a number of steps. (1) Each term k_i in the vocabulary is associated with an N-dimensional vector

$$\vec{k}_i = (w_{i,1}, w_{i,2}, \dots, w_{i,N}) \ , \qquad (14)$$

where N is the number of documents in the collection, and where $w_{i,j}$ is k_i's weight in the document d_j. How these weights are calculated is described in Baeza-Yates & Ribeiro-Neto 1999 (p. 132).

(2) The correlation $c_{u,v}$ between two terms, k_u and k_v, is calculated and a global *similarity thesaurus* (the collection-dependent knowledge structure) is generated. Here, the vectors of the terms are used. $c_{u,v}$ is defined according to

$$c_{u,v} = \vec{k}_u \cdot \vec{k}_v = \sum_{\forall d_j} w_{u,j} \times w_{v,j} \qquad (15)$$

Observe that all documents in the collection are used to find the correlation between the terms. The similarity thesaurus is obtained by calculating the correlation for each pair of terms in the vocabulary. We could place these values in a t times t matrix (t is the number of terms in the vocabulary). Generating the global similarity thesaurus demands a great deal of calculations. However, this only has to be done once, then the thesaurus is updated continually.

(3) This step expands the initial query q. (3.1) The similarity between each term k_v (which does not occur in q as we are going to expand q with new terms) and the initial query q is calculated by means of the similarity thesaurus according to

$$sim(q, k_v) = \sum_{k_u \in q} w_{u,q} \times c_{u,v} \ , \qquad (16)$$

where $w_{u,q}$ is the query term k_u's weight in the query q (assuming that the query terms are weighted, just like the terms in a document). Note that the addition is made across all query terms. A term with high correlation with highly weighted query terms gets a high similarity value.

(3.2) The terms are now ranked on the basis of the calculated similarity values. The *r* most highly ranked terms are chosen and added to the initial query *q*. The result is an expanded query q'. Then each expansion term k_v in the expanded query q' is weighted. Higher ranked expansion terms get higher weights than lower ranked ones. Finally, q' is matched against the document collection.

QE based on a collection-independent knowledge structure

Voorhees (1994) used a general knowledge structure, WordNet, as a source for expansion terms. WordNet is a manually constructed lexical system developed at Princeton University. The basic components of the system are sets of synonyms, *synsets*. These are organised by defining various lexical relations through them.

Voorhees used only the noun section of WordNet. Lexical relations with regard to nouns include the *is-a*-relation and three various *part-of*-relations. In Figure 2, a section of WordNet is shown, namely some of the relations that are defined for one of the meanings of the word "swing".[6] A rectangle represents a synset, the elements of which are listed in the rectangle. A link in the shape of a straight line represents the *is-a*-relation, and a link in the shape of an arch represents a *part-of*-relation. For example, a "trapeze" is a "swing", which in turn is part of a "playground".

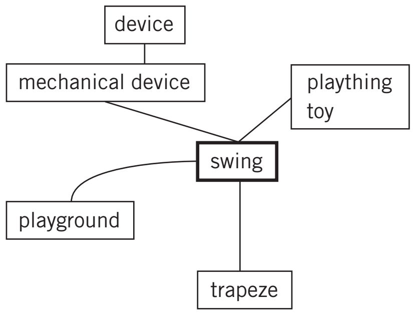

Figure 2. Some of the relations defined in WordNet for one of the meanings of the word "swing". The figure is based on Voorhees 1994 (p. 63).

Voorhees used a part of the TREC collection[7] in her experiments. 50 TREC topics were matched in each experiment against *ca.* 742,000 full text documents. The documents were automatically indexed, and the document terms weighted. The text in a TREC topic is a text in natural language, which expresses an information need (for an example of a TREC topic text, see Voorhees 1994, p. 64).

Voorhees examined, among other things, if an automatic choice of expansion terms from WordNet could improve the retrieval effectiveness. In the experiments in this partial study a query was derived, not from the whole text of a TREC topic, but only from a (summary) part of it. This resulted in initial queries with a small number of terms (compared with queries derived from the whole text of a TREC topic). The reason for Voorhees to experiment with short queries was the fact that another part of the study had failed to show that expansion of queries derived from the whole text of a TREC topic had improved the retrieval effectiveness (ibid., p. 65). The derived query terms were weighted, just like the terms of a document.

When the queries had been generated, the expansion itself could begin. The idea was for the system to choose terms, in an initial query, that were appropriate for expansion. Appropriateness was approximated with the number of documents in the collection in which the query term occurs (a term occurring in a great proportion of the documents in a collection is less useful with regard to distinguishing between relevant and non-relevant documents).

A chosen query term *k* could have several meanings. The system had to choose expansion terms in WordNet for *k* with meanings related to the meaning of *k* in the initial query. To achieve this, a term had to be related to at least two chosen query terms in order to qualify as an expansion term.

Let *q* be an initial query. The algorithm Voorhees applied in order to produce expansion terms for *q* could be informally described as follows. For each term *k* in *q*, the system first checks if the number of documents in the collection containing *k* is lower or equal to *N* (a threshold value).

If the number in question is higher than *N*, *k* will not be chosen. If the number is lower or equal to *N*, *k* will be chosen, and the system seeks each synset in WordNet that contains *k* and expands the set (i.e. follows links from the set to other synsets). Thereby, a list of potential expansion terms is created for each chosen term. The number of lists is then equal to the number of chosen terms in *q*. Finally, the system chooses each term that is found in at least two lists and these terms will form the expansion terms for *q*.

A number of experiments with automatically expanded queries were made, and the retrieval effectiveness (in terms of average precision[8] across all test queries) for the strategies of the experiments were compared to the effectiveness for the strategy of not expanding the initial queries.[9] None of the expansion strategies showed any improvement in effectiveness to speak of. Actually, the effectiveness decreased by 14.2 per cent for one of the strategies. One explanation for this poor result is that terms that occurred in two or more lists tended to be rather general and have more than one meaning (ibid., pp. 67f.).

Interactive QE

Efthimiadis (1995) studied eight ranking algorithms for QE. Examples of algorithms that were studied are *WPQ* and *Porter*, which are described above (see equations [1] and [2]). One of the aims of the study was to examine the effectiveness of the ranking algorithms regarding their ability to reflect users' preferences for expansion terms (ibid., p. 606). The users in the study were faculty members, researchers and doctoral students.

The study involved 25 searches, and each search involved one user. The initial queries were matched against a database consisting of bibliographical records. The matching process resulted in a ranked list of records, and the user judged a number of top ranked records for relevance. All unique terms from the descriptor and identifier fields in the records judged relevant by the user were presented to him. He then chose the terms he considered to be useful as expansion terms, and ranked them.

The term weights (for all unique terms from the descriptor and identifier fields in the records judged by the user to be relevant) for each algorithm and each search were then calculated, and the terms ranked on the basis of the weights. For a given search, eight ranked term lists were generated, one for each algorithm.

In order to establish whether or not the term lists from the eight algorithms reflected the preferences of the users with regard to expansion terms, Efthimiadis experimented with, among other things, rank sums. For a given search, each term from the lists was given a rank. Thereby, it was possible to find out which ranks the user's five most highly ranked terms had in the various lists. Then, the sum of the five ranks was calculated for each list.[10] For a given search, eight rank sums were generated, one for each algorithm. The lower a rank sum an algorithm had in a given search, the better it reflected the user's choice with respect to the five best terms. The mean values (over all 25 searches) with regard to the rank sums of the algorithms were calculated. It was shown that *WPQ*, together with another algorithm, had the best result, i.e. the lowest mean value. *WPQ* approximated the users' preferences well in comparison with most of the other algorithms.

It is important to remember that the point, against which the effectiveness of the various algorithms was measured, was the users' choice of best terms. Assume that a certain algorithm is used for *automatic* QE. If users prefer inappropriate terms (from a retrieval point of view), it is hardly expedient that the algorithm places preferred terms high on the ranking lists (i.e. is effective in the sense used in Efthimiadis's study). This would mean that the initial query is expanded in an inappropriate way.

If we assume, instead, that users generally choose terms for expansion in a rational way – which may be a reasonable assumption, at least concerning sophisticated information seekers – a ranking algorithm that places preferred terms high on the ranking lists, could be convenient in interactive (and automatic) QE. As opposed to the system presenting unranked lists of terms from documents judged to be relevant (lists from which the user is supposed to choose terms for expansion), the terms from documents judged to be relevant are presented, with the aid of the algorithm, in ranked lists, with regard to appropriateness (from the system's point of view). Users will then quickly find terms that they prefer (and that are suitable for expansion), since such terms tend to be found high in the ranked lists.

It could be mentioned that Efthimiadis also used the rank sums of the algorithms for another purpose. Apparently, two sets of rank sums were generated for each pair of algorithms *A* and *B*, one for *A* and one for *B*, with 25 rank sums in each. In order to find out if *A* and *B* behaved in a similar way across the 25 searches (and with respect to the user's five best terms), Efthimiadis applied the correlation measure *Person's r* on the two sets of rank sums. A very strong positive correlation was observed between several algorithms. A positive correlation means, for two given algorithms *A* and *B*, that a high rank sum for *A* generally corresponds to a high rank sum for *B* and that a low rank sum for *A* generally corresponds to a low rank sum for *B*. In fact, a positive correlation was found for each pair of the studied algorithms.

Concluding remarks

It is not unusual that users of WWW search engines construct queries consisting of a small number of terms (Baeza-Yates & Ribeiro-Neto 1999, p. 13). Such queries can be very ineffective but could be improved by applying QE. The three latter methods for automatic QE described above are attractive in this context because users do not have to perform any relevance judgements. They only need to type the initial query. The system then expands the query, matches the expanded query against the document collection and presents a search result to the user.

To construct an automatic procedure for choosing expansion terms from a knowledge structure of WordNet's kind is no trivial task. As seen above, Voorhees's automatically expanded queries did not improve the retrieval effectiveness to any greater extent. The terms chosen by the system to be added to initial queries tended to be ambiguous. Such terms match documents in which the term has been used in a sense that deviates from the sense that is relevant for the query. This may result in the retrieval of non-relevant documents, which has a negative impact on precision. However, the idea of using a lexical system of WordNet's kind might be fruitful. It would be interesting to use a Swedish equivalent to WordNet for QE. In that case, it might be convenient to find another process for automatic choice of terms from the lexical system than the one used by Voorhees.

The method of QE by local clustering runs into problems if none or few of the initially retrieved documents are relevant. Assume that no retrieved document is relevant. Under this assumption, it is hardly advisable to expand the initial query with terms from local association clusters. The problem is that there is no support for the assumption that a term, belonging to a local association cluster around a query term k_u, would be effective in retrieving relevant documents. Such a term co-occurs with k_u in (initially retrieved) non-relevant documents.

Per Ahlgren is a Ph.D. student at the Swedish School of Library and Information Science, University College of Borås/Göteborg University. His main interests are information retrieval and bibliometrics.
E-mail: Per.Ahlgren@hb.se

Notes

1. A *query* is a formal representation (in a given IR system language) of an information need.

2. An IR model could be said to be a simplified (mathematical) theory of how an IR system should be constructed.

3. The weights consist of numerical values and quantify the importance, with respect to description of semantic content, that the corresponding term has in the document or the query, respectively.

4. If this is true, the term will be absent in the new query.

5. The claim that the method is an example of QE is slightly improper. As is shown above, the method could involve deletions of initial query terms.

6. The word "swing" has several other meanings. Also for its other meanings, different lexical relations are defined in WordNet.

7. TREC (Text Retrieval Conference) is an on-going series of retrieval experiments, involving research groups from various parts of the world. See Text REtrieval Conference 2001.

8. The fraction of the retrieved documents which are relevant.

9. In Voorhees's study, both documents and queries were represented by vectors. In the various experiments of the study, the similarity between document vectors and query vector was measured, and the documents were ranked on the basis of the measured values. The various expansion strategies differed, among other things, concerning the value of *N*.

10. If the five terms are found in the first five positions in the list of an algorithm, the optimal rank sum is achieved, namely 15.

References

BAEZA-YATES, R. & RIBEIRO-NETO, B. (1999): *Modern Information Retrieval*. New York: ACM Press.

EFTHIMIADIS, E. (1995): User Choices: A New Yardstick for the Evaluation of Ranking Algorithms for Interactive Query Expansion. *Information Processing & Management*, no. 4, vol. 31, 605–620.

EFTHIMIADIS, E. (1996): Query Expansion. In: M.E. Williams, ed. *Annual Review of Information Science and Technology*, vol. 31. Medford, N.J.: ASIS, 121–187.

HARTER, S. (1986): *Online Information Retrieval: Concepts, Principles, and Techniques*. San Diego: Academic Press.

KORFHAGE, R. (1997): *Information Storage and Retrieval*. New York: John Wiley & Sons.

TEXT RETRIEVAL CONFERENCE (2001): *Text REtrieval Conference (TREC) HomePage*. URL: http://trec.nist.gov. [2001-04-19]

VOORHEES, E. (1994): Query Expansion Using Lexical-Semantic Relations. In: W.B. Croft & C.J. van Rijsbergen, eds. *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3–6 July 1994*. ACM/Springer, 61–69.