A Deductive Data Model for Query Expansion

Kalervo Järvelin, Jaana Kristensen, Timo Niemi#, Eero Sormunen and Heikki Keskustalo
Dept. of Information Studies, #Dept. of Computer Science
University of Tampere
P.O.Box 607 FIN-33101 TAMPERE, Finland
INTERNET: {likaja, lijakr, tn, lieeso, ccheke}@uta.fi
FAX: +358 - 31 - 2156 560 TEL: +358 - 31 - 2156 111

Abstract: We present a deductive data model for concept-based query expansion. It is based on three abstraction levels: the conceptual, linguistic and occurrence levels. Concepts and relationships among them are represented at the conceptual level. The expression level represents natural language expressions for concepts. Each expression has one or more matching models at the occurrence level. The models specify the matching of the expression in database indices built in varying ways. The data model supports a concept-based query expansion and formulation tool, the ExpansionTool, for heterogeneous IR system environments. Expansion is controlled by adjustable matching reliability.

1. INTRODUCTION

Thesaurus modeling and software have received much attention in information retrieval (IR) literature. Jones & al (1993; 1995) present a thesaurus data model, based on the relational data model (RDM) and investigate the feasibility of incorporating intelligent algorithms into software for thesaurus navigation. A thesaurus database can also be used for automatic query expansion (QE) whereby a query is reformulated by adding new terms provided by the thesaurus.

QE can be performed, on one hand, prior to the initial search or the relevance feedback search and, on the other hand, on the basis of statistical or linguistic information (Ekmekcioglu & al., 1992) The source of expansion may be document titles, a thesaurus, or a classification (Hancock-Beaulieu, 1992). Thesaurus-based QE may be performed through a spreading activation method (Paice, 1991) or through ordinary thesaural relationships (e.g., association and hierarchical relationships, Kristensen, 1993).

Ekmekcioglu (& al., 1992) found that statistical or linguistic QE did not provide significant difference in retrieval effectiveness when compared to unexpanded queries (ranked output, over 26 000 title-and-abstract documents) Kristensen (1993) reported a doubling in recall with a 11 % decline in precision (63 to 51 %) for thesaural QE (Boolean retrieval, 227 000 newspaper articles). Jones & al (1995) found that query expansion through a thesaurus in a ranked output system reordered the result but did not improve it (ranked output, large INSPEC database)

SIGIR'96, Zurich, Switzerland©1996 ACM 0-89791-792-8/96/08.$3.50

In this paper we present a new approach by providing a deductive data model for thesauri because the ordinary RDM does not support transitive relationships typical in thesauri. If some term B is an immediate narrower term (NT) of another term A and the term C an NT of B, then C is also, transitively, an NT of A, i.e., A a is broader term (BT) of C NTs and BTs of given terms are frequently needed in thesaurus navigation and QE. This requires the computation of the transitive closure (Ullman, 1989) among the relationships and starting terms.

In the RDM computation of the transitive closure is practically impossible but in deductive databases it is possible. In the latter, transitive relationships form the most frequent recursion type for rule-based computation (Chang & Walker, 1986). Our work has this starting point. Agrawal's (1987) extended relational algebra and our deductive query language (Niemi & Järvelin, 1992) are operation-oriented approaches to transitive queries. Our approach provides a set of specific operations instead of one transitive closure operation Especially its direction-specific operations are valuable in thesaurus navigation Through them we can find BTs or NTs of given terms by specifying the direction. Our approach supports the computation of transitive relationships in a collection of binary relations This is very useful because thereby QE can be restricted to particular types of concepts (e.g., persons or things) and particular types of relationships (e.g , generic or partitive relationship)

In our thesaurus data model, deductive operations are used to manage hierarchic relationships, i.e , generic, partitive and instance relationships. They are not used to manage equivalence nor associative relationships The former are rather relationships between concept expressions than between concepts and are managed in the RDM. The latter are non-directed, reciprocal concept relationships and thus cannot be managed by our deductive operations which presently assume acyclic binary relations. However, this is not a limitation since the associative relationship performs badly if utilized transitively in query formulation.

In our deductive data model the *conceptual level* represents concepts and conceptual relationships. The *linguistic level* represents natural language expressions for concepts Typically there are many expressions of varying reliability for each concept Each expression may have one or more matching models of varying reliability at the *occurrence level*. Each matching model represents, in a query-language independent way, how the expression may be matched in texts or database indices built in varying ways, e.g., with or without stemming and with or without compound words split into component words.

We shall apply the deductive data model for QE and present a tool, called the ExpansionTool, for parametrized automatic QE. This tool is intended for QE

- prior to the initial search
- for varying search topics and search exhaustivity (e.g , high recall vs. high precision)

- based on a searching thesaurus (or a semantic expansion approach)
- for natural language text retrieval in document collections lacking intellectual indexing
- in heterogeneous retrieval environments where the database index types, retrieval systems and matching methods (Boolean or term-weighted, ranked retrieval) vary; thus it must preserve query structure for retrieval systems which may utilize it.

Currently the ExpansionTool assumes that the user already knows the concepts relevant to her/his search. The ExpansionTool then expands the concepts and constructs a query.

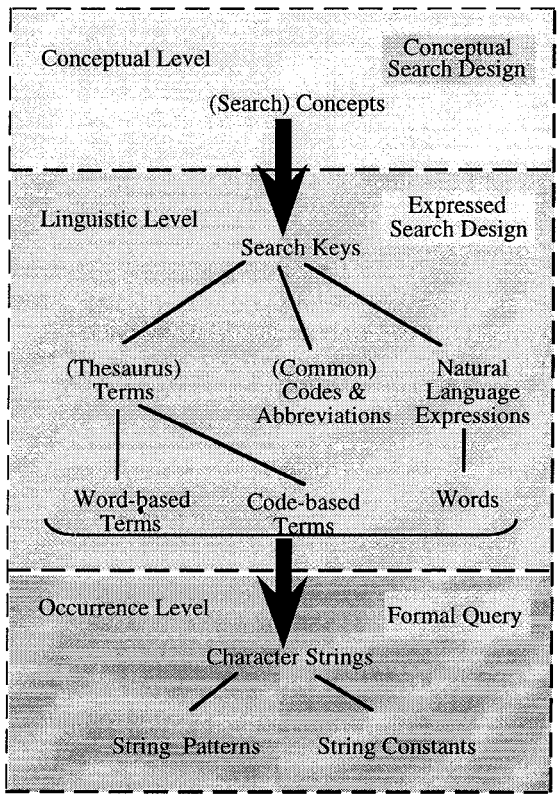

Fig. 1. The abstraction levels of query formulation

2. THE THESAURUS DATA MODEL

2.1. Three abstraction levels

The three abstraction levels: conceptual, linguistic and occurrence level are well-founded in the IR literature (Croft, 1986; Paice, 1991; UMLS, 1994). Thus we can differentiate between concepts and their structures (the generic, partitive and associative relations) at the conceptual level, concept expressions and their structures (the equivalence relation) at the linguistic level, and matching models (e.g., full-word strings, stems, string patterns containing wild cards) indicative of linguistic expressions at the occurrence level. Expressions represent concepts and each concept may have several expressions in several natural and artificial languages. The expressions may be basic words, compound words, phrases or larger linguistic constructs, or common codes and abbreviations (e.g., USD49.90). Our approach is intended for QE where the occurrence level contains matching models for matching expressions in text, and where confidence figures between concepts and their possible expressions as well as between expressions and their matching models are utilized in query construction.

Figure 1 illustrates the roles of the three levels in query formulation. Search concepts are first translated into search keys which are (thesaurus) terms, common codes and/or natural language expressions. Thereafter the search keys are translated into matching models, e.g., string patterns with wild cards or string constants. Language-dependent aspects are represented at the linguistic and occurrence levels. All retrieval system dependent aspects are encapsulated at the occurrence level in system-specific translators.

2.2. Thesaurus representation

Representation of concepts, expressions and matching models

We represent concepts, their linguistic expressions and matching models in a relational database illustrated in Figs. 2-5. Concepts are represented in the relation **CONCEPTS** which has the attributes **CNO** (concept number), **CNAME** (concept name), **CATEGORY** (concept category), and **DEFINITION** (concept definition). Concept categories are chosen according to the application domain of the thesaurus. For example, in a thesaurus for newspaper articles, the concept categories might include persons, organizations, things and events.

CONCEPTS			
CNO	**CNAME**	**CATEG.**	**DEF'N**
c1	forest industry	org	...
c2	chem forest industry	org	...
c3	mechn forest industry	org	...
c5	paper industry	org	...
c7	plywood industry	org	...
c10	paper company	org	...
c12	Kymi-Kymmene	org	...
c100	investment	proc_event	...
c300	protection	proc_event	...

Fig. 2. The relation **CONCEPTS**

Expressions are represented in the relation **EXPRESSIONS** with two attributes **ENO** (expression number) and **EXPRESSION** (the expression). The relationships between concepts and their expressions are given in the link relation **CONS_EXPRS** which has the identifier attributes **CNO** and **ENO**, as well as the attributes **ETYPE** (expression type) and **STRENGTH**. The attribute **ETYPE** defines the expression as a preferred term for the concept (term), a synonym (syno), or other type included in the ISO standard (ISO, 1986) for thesaurus construction. The **STRENGTH** attribute specifies the strength of the association between the concept and its expression as a real number between [0, 1]. The source of strength figures can be human judgment, statistics based on the meanings of expressions, or relevance feedback. Fidel and Efthimiadis (1995) point out the need for terminological research to analyze such attributes.

CONS_EXPRS			
CNO	**ENO**	**ETYPE**	**STRENGTH**
c1	e1	term	1.0
c1	e2	syno	1.0
c100	e100	term	1.0
c100	e101	verb	0.8

Fig. 3. The relation **CONS_EXPRS**

The matching models are represented in the relation **OCCURRENCES**. It has three attributes **ONO** (occurrence number), **OTYPE** (occurrence type), and **OCCURRENCE**. The occurrence type indicates whether the **OCCURRENCE** attribute gives a matching model for a morphological basic word form (bw) or for a stem of inflected forms (st). The idea here is that the database index may contain either morphological basic forms recognized by, e.g. the TWOL software (Koskenniemi, 1983), or inflected words as they appear in the text. The matching model representation is retrieval system independent and has the following features:

EXPRESSIONS	
ENO	**EXPRESSION**
e1	forest industry
e5	paper industry
e11	paper company
e12	Kymi - Kymmene
e121	Kymi ltd
e100	investment
e101	to invest

EXPR_OCCS		
ENO	**ONO**	**RELIAB.**
e1	o1	0.9
e1	o112	0.9
e70	o70	0.9
e100	o100	1.0
e101	o101	1.0
e500	o500	0.6
e500	o501	0.9

Fig. 4. The relations **EXPRESSIONS** and **EXPR_OCCS**

OCCURRENCES		
ONO	**OTYPE**	**OCCURRENCE**
o1	bw	prox(<bw(forest), bw(industry)>, 3)
o70	bw	adj(<cw(<bw(ply), bw(wood)>), bw(industry)>, 4)
o100	bw	bw(investment)
o101	bw	bw(invest)
o112	st	phra(<bw(forest), st(industr)>)
o501	st	st(factor)

Fig. 5. The relation **OCCURRENCES**

- Representation of atomic words by bw(*word*), when *word* is a morphological basic form (e.g., bw(investment)), and st(*stem*), when *stem* is a stem (e.g., st(factor)).
- Representation of compound words by their morphological basic forms or stems. The basic form matching models represent compound word components because they may be represented in the database index. In many languages, the component words may occur in inflected forms in the compound but are in the basic form if split. Thus the compound word basic forms are expressed by cw(<*c1, ..., cn*>), when *c1, ..., cn* are component words in the correct order. Components which may inflect in the compound are modeled by iw(*basicform, inflform*), where *basicform* and *inflform* are the basic and inflected forms, respectively. Other components are modeled by the basic word form construct. For example, "plywood" is modeled by cw(<bw(ply), bw(wood)>).
- Phrase matching with a defined word order through morphological basic forms or stems by the model phra(*Comps*), where *Comps* gives the component words. The components may be stems, basic words or compound words. For example, "forest industry" can be represented by phra(<bw(forest), st(industr)>).
- Matching of words in defined and undefined order, with intervening words allowed, through morphological basic forms or stems. The models are prox(*Comps, Distance*) and adj(*Comps, Distance*), where *Comps* gives the components and *Distance* gives the allowed distance between components. For example, "forest industry" may be modeled by prox(<bw(forest), bw(industry)>, 3) in this order and with distance of 0 - 3 words.

The relationships of expressions and occurrences are given in the link relation **EXPR_OCCS** which has the identifier attributes **ENO** and **ONO**, as well as the attribute **RELIABILITY**, which gives the matching model's reliability for the intended expression as a real number between [0, 1].

The occurrence level is necessary for several reasons. First, document texts may have basic words, compound words and phrases which may or may not have been analyzed into their morphological basic forms for the database index. Recognition of compound word components would require a linguistic analysis program within Expansion-Tool. Second, each expression may have several matching models, each of varying reliability. Third, the reliability of a matching model requires human judgment. Fourth, even if the solution of the judgmental problems could be mimicked to a sufficient degree by some statistical techniques, the knowledge represented as matching models at the occurrence level would have to be constructed each time an expression is needed for a query. Thus query language-specific queries are not translated from the expressions directly but from the matching models as shown below.

The characteristics of matching models are necessary when the document texts may have basic words, compound words and phrases. Compound words are frequent in many European languages. For example, the German word Verkehrswegeplanungsbeschleunigungsgesetzveränderungsentwurf makes clear that by splitting compound words for the index, hidden components may become retrievable, e.g. gesetz + veränderung.

The matching models extend the ideas developed in the I^{3}R system (Croft, 1986; Croft & Das, 1990) by taking possible index types into account and by providing system independent support to query generation. The matching models are not used to provide evidence of document content but, instead, to generate queries.

Representation of concept relationships

Concept relationships are either association relationships or hierarchic relationships. They are always non-synonymous relationships because the latter are relationships between expressions. The ISO thesaurus construction standard (ISO, 1986) gives examples of types of association and hierarchic relationships usable in the data model.

ASSOCIATIONS			
CNO	**ASS_CNO**	**ASS_TYPE**	**ASS_STRENGTH**
c1	c10	sibling	0.8
c10	c1	sibling	0.8
c100	c500	process	0.4
c500	c100	process	0.4

Fig. 6. The relation **ASSOCIATIONS**

The relation **ASSOCIATIONS** (Fig. 6) represents the association relationships between concepts through the attributes **CNO** and **ASS_CNO**, containing pairs of associated concepts, as well as the attributes **ASS_TYPE** (association type) and **ASS_STRENGTH** (association strength). The association type value specifies how the concepts are associated, e.g., sibling concepts (industry - company) and process relationship (investment - factory) (ISO, 1986).

Representation of hierarchic relationships is based on binary relations partitioned according to concept categories (e.g., organizations and processes) and hierarchy types (e.g., generic and partitive relationships). The categories

and relationship types can be chosen according to the thesaurus domain. For example, our sample thesaurus database contains the binary relations of Fig. 7 where organizations have a binary relation for the generic relationship and another for the instance relationship, processes and events have a binary relation for the generic relationship, etc.

ORG_GENERIC	
BROADER_CONS	**NARROWER_CONS**
c1 (forest industry)	c2 (chem forest ind)
c1 (forest industry)	c3 (mechn forest ind)
c2 (chem forest ind)	c5 (paper industry)
c3 (mechn forest ind)	c7 (plywood indust)
c9 (wood proc comp)	c10 (paper company)

PROC_EVENT_GENERIC	
BROADER_CONS	**NARROWER_CONS**
c100 (investment)	c101 (environ invest)
c100 (investment)	c102 (factory invest)
c101 (environ invest)	c103 (water clean inv)
c101 (environ invest)	c104 (air protect inv)
c101 (environ invest)	c105 (forest protect i)

Fig. 7. The relations **ORG_GENERIC** and **PROC_EVENT_GENERIC**

The advantages of partitioning the hierarchical relationships are: (i) efficiency, because the computation of transitive relationships is faster in smaller binary relations, and (ii) precision, because concepts can be analyzed and expanded in controlled relationships.

3. CONCEPT-BASED QUERY EXPANSION

3.1. Expansion principles and parameters

The ExpansionTool expands conceptually represented queries at the three abstraction levels. When the necessary matching models at the occurrence level have been retrieved, they are translated into the specified target query language. The ExpansionTool is an application software implemented in Prolog. It utilizes, internally, queries of the integrated deductive query language by Niemi & Järvelin (1992). Järvelin (& al., 1996) contains definitions of the functions for performing various expansions.

We shall use a request on environment protection investments of paper and mechanical forest industry as our expansion example. Assume further, that the user has identified the concepts c5 and c3 suitable in describing the industries and c101 suitable in describing the investments.

Concept expansion

The starting point of concept expansion is a set of concept sets interpreted as conjunctive facets representing the information need. Within each set, the concepts are alternative (or disjunctive) interpretations of the facet. Thus the starting point is a set of concept facets C = {F_1, F_2, ... , F_k}. In our sample case it is C1 = {{c3, c5}, {c101}} where {c3, c5} and {c101} are the facets of concept identifiers. In principle, there is an "AND" between the facets and an "OR" between the concepts within each facet, e.g., between c3 and c5. This conjunctive normal form (CNF) structure is maintained throughout query construction and rejected only in the translation phase if the retrieval setting requires it.

The expansion principle is that each concept is expanded to a disjunctive set of concepts on the basis of conceptual relationships pointed out by the user. If expansion to broader concepts is requested, only the immediate broader concepts are added, because otherwise the meaning of the query would probably expand too far. If expansion to narrower concepts is requested, the concepts are expanded to all immediate or indirect narrower concepts. If associative expansion is requested, expansion is by one link in the association relationship. In the hierarchical relationship the user has a choice of hierarchical relationship type. One or more may be chosen. If expansion to instance concepts is requested, the instance concepts of all expanded concepts are included. Each relationship type is applied once for expansion. In other words, concepts like associated concepts of narrower concepts (or vice versa) are not computed.

The expansion parameters are (1) set of expansion type indicators for concept expansion and (2) the required reliability. The former is a set of relationship names, *bcg* indicating generic broader concept relationship, *bcp* partitive broader concept relationship, *ncg* generic narrower concept relationship, *inst* the instance relationship, *asso* the associative relationship, etc. Both parameters are given to each facet separately.

The expansion result is a set of expanded concept facets {F_1', F_2', ... , F_k'} where each facet F_i' contains the original and the expanded concept identifiers. Thus each facet is expanded separately. In the sample case the industry facet is expanded by *bcg, ncg* and *inst* in the ORG_GEN and ORG_INST relationships, and the investment facet by *bcg, ncg* and *inst* in the PROC_EVENT_GEN relationship. The reliability figures are 0.8 and 0.6, respectively. The resulting expanded concept identifier facets are C1 = {{c5, c2, c3, c1, c7, c8}, {c101, c100, c103, c400}} where c5 = paper industry, c3 = mechanical forest industry, c1 = forest industry, c7 = plywood industry, c101 = environment investment, c100 = investment, c103 = waste water treatment investment, c400 = environment, etc.

Expression expansion

Expression expansion starts with the concept expansion result {F_1', ... , F_k'}. The expansion principle is that, for each concept identifier, identifiers of selected expressions are collected (ORed) as representatives. The expansion parameter is a reliability figure guiding the selection of expression identifiers. All identifiers with strength exceeding the figure are collected for each concept. The result is an expanded set of expression facets {E_1', ... , E_k'}, where each facet E_i is derived on the basis of the corresponding concept facet F_i.

The expression expansion for C1 (above) returns the set E1 = {{e50, e4, e5, e30, e1, e2, e70, e80}, {e1010, e100, e101, e103, e4001}}. Here, for example, e50 = paper industry (for concept c5), e30 = mechanical forest industry (for c3), e1010 = environmental investment (c101), e103 = waste water treatment investment (c103) and e4001 = biosphere (c400).

Occurrence expansion

The starting point of occurrence expansion is a set of expression identifier facets E = {E_1, ... , E_k} constructed above. The occurrence expansion principle is to collect, for each expression identifier, all its matching models which are reliable enough and suitable for a database index of a given type specified as expansion parameters. The occurrence expansion result is a set of matching model facets O = {O_1, ..., O_k} where each facet O_i is derived from the corresponding expression identifier facet E_i by replacing each

expression by a set of matching models. The sample expression expansion result E1 yields the set of matching model facets O1 =

```
{{ prox(<bw(paper), bw(industry)>, 2),
   prox(<bw(chemical), bw(forest), bw(industry)>, 3),
   prox(<bw(chemical), bw(wood), bw(processing),
            bw(industry)>, 3),
   prox(<bw(mechanical), bw(forest), bw(industry)>, 3),
   prox(<bw(forest), bw(industry)>, 3),
   prox(<bw(wood), bw(processing), bw(industry)>, 3),
   prox(<cw(<bw(ply), bw(wood)>), bw(industry)>, 3),
   prox(<cw(<bw(saw), bw(mill)>), bw(industry)>, 3)},
 {  prox(<bw(environmental), bw(investment)>, 2),
   bw(investment), bw(invest),
   prox(<bw(waste), bw(water), bw(treatment),
            bw(investment)>, 2),
   bw(biosphere) }}
```

when the database index type is set for basic word forms and the reliability threshold is 0.8 for the industry facet and 0.6 for the investment facet.

Matching model translation

This step translates the query language independent expression into a query of a given language. The starting point of matching model translation is the expansion result constructed above. Matching model translation is implemented in the basis of logic grammars (Abramson & Dahl, 1989). Each grammar is a set of logical rules which generate well formed expressions of a specified query language. Each query language has its own logic grammar which takes its expression types and syntax limitations into account.

Usually logic grammars must be translated into logic programs separately, e.g., the rules of the well-known logic grammar DCG (Definite Clause Grammars, Pereira & Warren, 1980) are translated into normal Prolog clauses by adding extra arguments to each non-terminal. Because Prolog as such supports parsing of functional expressions we constructed our logic grammar directly as an executable Prolog program. The heads of the rules of our logic grammars consist always of two components: the standard source language component and the target language component. By using these components we parse and generate functional expressions at the same time using the one-pass approach.

The parameters of matching model translation are (1) the facet operator, (2) the database index type indicator and (3) the target query language identifier. The first one is used to express whether the facets are combined by a disjunction (the operator OR), conjunction (AND), or a paragraph (PARA) or a sentence (SENT) proximity condition, or by a probabilistic sum (the operator SUM). The database index type indicators bw, cw and iw indicate index types "basic words with compound words split", "basic words with compound words not split" and "inflected words", respectively. Allowed query language identifiers currently are one of *inquery* (for INQUERY v1.6 by University of Massachusetts), *iso* (for the ISO standard query language; ISO, 1993), *topic* (for TOPIC by Verity Inc.), or *trip* (for TRIP by PSI Inc.).

If the target language of matching model translation does not support some specific feature of matching models or logical structure, then either the obvious closest or alternative construct of the target language is generated or query construction terminates with an error message. For example, the INQUERY retrieval system (v1.6) does not have grammatical proximity operators (e.g., "sentence") but supports proximity conditions based on numeric word distance. Therefore the sentence proximity condition is translated a numeric proximity expression #10 allowing 10 intervening words. INQUERY neither supports disjunctions within proximity operations, i.e., #10(#or(a, b), #or(d, e)). Therefore such structures are automatically converted into DNF, i.e. #or(#10(a, d), #10(a, e), #10(b, d), #10(b, e)) which is supported. The TOPIC query language provides the proximity operators "phrase", "sentence" and "paragraph". Proximity models cannot be modeled by the "phrase" operator and thus need the next weaker operator "sentence". If the facet operator is also "sentence", then the grammar avoids constructing a nested sentence operation sequence. All such transformations are handled by the logic grammars.

The result of matching model translation is a query of the target query language for an index of the specified type. The query may be very long, if it contains many broad concepts expanded by loose criteria, and if a proximity condition is applied between the facets.

We shall outline the matching model translation algorithm briefly. In the algorithm *mm-trans(mm, it, G)* denotes the translation of the matching model *mm* for an index of type *it* by the logic grammar *G*. Let G' be the logic grammar for INQUERY. An example of a recursive translation rule is:

mm-trans(prox(<C1, ... , Cn>, Dist), it, G') →
#Dist(*mm-trans*(C1, it, G'),... , *mm-trans*(Cn, it, G'))

which translates a proximity model recursively into INQUERY expressions. The matching model translation algorithm is outlined as follows:

Algorithm TRANSLATE
Input: *TQL*, the target query language identifier
FOp, the facet operator
it, the target index type
MF, the matching model facet set
Output: *Q*, a query in the query language identified by *TQL*
Procedure:
Step 1. Identify the required logic grammar *G* through the target language identifier *TQL*.
Step 2. Select an appropriate strategy on the basis of *FOp:*
Case I. *FOp* = AND. Translate *MF* in CNF by *G*. Apply *mm-trans(mm, it, G)* to each matching model *mm*.
Case II. *FOp* = PARA or *FOp* = SENT. If *TQL* does not support CNF in proximity operations, turn *MF* into *MFD* (representing *MF* in *DNF*) and translate *MFD* by *G* by applying *mm-trans(mm, it, G)* to each matching model *mm*. Otherwise translate *MF* in CNF by *G* by applying *mm-trans(mm, it, G)* in the same way.
Case III. FOp = OR. Translate MF as a two-level disjunctive structure by *G*. Apply *trans(mm, it, G)* to each matching model *mm*.
Case IV. *FOp* = SUM. Translate MF as a three-level probabilistic sum structure by *G* as explained below. Apply *trans(mm, it, G)* to each matching model *mm*.

In case IV, the most reliable model, say M_1, for the original unexpanded concept in each facet is separated from the rest, say $\{M_2, ..., M_n\}$ and they are translated and combined as a two-level subquery (S_1 OR SUM(S_2, ..., S_n)), where S_i = *trans*(M_i, *it, G*) when i = 1,..., n. The subqueries are combined linearly by the SUM-operator, i.e., as SUM((S_{11} OR SUM(S_{12}, ..., S_{1n})), (S_{21} OR SUM(S_{22}, ..., S_{2n})), ...).

We shall consider sample queries generated for the INQUERY and TRIP retrieval systems. INQUERY allows ordinary Boolean as well as probabilistic retrieval by Boolean operators '#and', '#or', and '#not', and proximity searching

by the operator '#*n*', where *n* is an integer. The proximity operator '#*n*' spans over sentence and paragraph boundaries. The probabilistic sum operators '#wsum' and '#sum' are also available. The retrieval system TRIP allows ordinary Boolean searching ('and', 'or', 'not') and proximity searching by operators 'and.p', 'and.s' and ' . . ' for paragraphs, sentences and phrases, and string matching by several types of wild cards. For phrases, the number of periods indicates the number of allowed intervening words. TRIP interprets full stops as sentence delimiters.

The matching model facet set O1 translates into the following TRIP query when the facet operator is SENT and the database index type is *iw* for inflected word forms:

```
(paper . . industr# or forest# . . . industr#
or wood# . . . process# . . . industr# or
plywood . . . industr# or sawmill# . . .
industr#) and.s (invest# or biospher#)
```

The two facets are combined by the operator **and.s** and character strings generated from the matching models by **or**. The proximity matching models have generated character strings containing the TRIP proximity operator ". ." for specified word order and allowed distance given in the matching models. The stem matching models have generated character strings containing wild cards. Expressions like "chemical# . . . wood . . . process# . . . industr#" are not generated because "wood . . . process# . . . industr#" matches it. The logic grammar deletes all logically redundant matching models.

The translation of the same request into the INQUERY query language with paragraph proximity and a *bw* type of index yields the result (operators in bold face):

```
#or(#20(#2(paper, industry), investment),
   #20(#2(paper, industry), invest),
   #20(#2(paper, industry), biosphere),

   #20(#3(forest, industry), investment),
   #20(#3(forest, industry), invest),
   #20(#3(forest, industry), biosphere),...)
```

The three dots represent three more blocks with **#3**(wood, processing, industry), **#3**(**#0**(ply, wood), industry), and **#3**(**#0**(saw, mill), industry) instead of **#2**(paper, industry) and **#3**(forest, industry). The query is considerably longer than the previous one. This is due to the DNF form needed in INQUERY proximity operations. The facet operator *para* was translated by **#20**. The other proximity expressions **#2** and **#3** have the distance given in the matching models. Because the target index splits compound words, the compound words plywood and sawmill would be matched by **#0**(ply, wood) and **#0**(saw, mill), if they were not deleted as redundant. The operator **#0** indicates that the component words must have the same address in the index.

The translation of the same request into the INQUERY query language with the probabilistic sum operator and a *bw* type of index yields the result:

```
#sum(#or(#2(paper, industry),
       #sum(#3(forest, industry),
            #3(wood, processing, industry),
            #3(#0(ply, wood), industry),
            #3(#0(saw, mill), industry))),
     #or(investment,
      #sum(invest, biosphere))).
```

3.2. The Expansion Tool

The ExpansionTool performs automatic QE in the way outlined above. The adjustable parameters of the current implementation are:

- A concept facet specification. Each facet contains one or more disjunctive concepts, required expansion types (bcg, ncg, asso, ...) and a reliability figure [0.0 ... 1.0].
- The facet operator (or, and, para, sent, sum).
- The database index type indicator (bw, cw, iw).
- The target IR system indicator (inquery, iso, topic, trip)

The current implementation uses only one reliability figure for concept expansion, expression identification, and matching model collection. The ExpansionTool is implemented in Prolog. Thus the tool generates automatically several query versions in sequence, if one or more of the last three parameters are given as variables.

3.3. Test of Expansion Effects

The test seeks to show that the ExpansionTool can be used for automatic QE with desirable effects on query results with respect to recall and precision.

The test environment

The test text database contained about 54.000 newspaper articles published in three Finnish newspapers in 1988-1992. The whole database (125 MB) contained some 12.5 million words. The environment also provides 35 search requests and for each the original verbal requests, their conceptual analyses, and collections of possible expressions for each concept. The relevance of altogether some 8000 documents is known for the 35 search requests. This relevance base was retrieved for assessment by a combination of a high-recall Boolean query and several probabilistic queries for each search request.

The database index for INQUERY contained all keys in their basic forms with compound words split. Morphological analysis was performed by the TWOL software (Koskenniemi, 1983). The database index for TRIP contained all word occurrences in their inflected forms.

The test thesaurus was in Finnish for a Finnish database and contained:

- 381 concepts
- 619 expressions for the concepts
- 1346 matching models for the expressions.

The test thesaurus contained concepts, expressions and matching models (as described above) for the topics present in ten test queries on: (1) the Bush - Gorbatchov Summit in Helsinki in 1991, (2) the South-American debt crisis, (3) dumping charges against Finnish forest industry in the US. (4) annihilation of Iraqi mass destruction weapons, (5) revolts in Bucharest against opposition by miners whom President Iliescu called to help the government, (6) the UN peace protection operation for Namibia's independence, (7) The 2 + 4 negotiations between East and West Germany and the four allied concerning the union of Germany, (8) the processing and storage of waste from nuclear power plants, (9) the effects of import restriction removal on Finnish food processing industry, (10) packages as an environment protection issue.

Thesaurus growth would affect slightly the test results. Conceptual growth in the thesaurus does not add expressions or matching models to concepts already in the thesaurus but may bring new associations. No matter how the thesaurus grows, a concept is expressed and matched exactly in the same way unless its conceptual relationships, expres-

sions or matching models are modified explicitly. Thesaurus growth has significant effects only if the thesaurus is used so that search requests are mapped (through their words) automatically to concepts which then are expanded.

Expansions

Queries were constructed by the ExpansionTool and tested based on the following expansion types (figures in parentheses indicate reliability threshold values employed to filter the relations):

- os the original query (words from the original search request) provides the baseline comparison data
- bt, bl basic synonym query, tight (bt, 0.89) and loose (bl, 0.49) interpretation
- nt, nl narrower concept expanded query, tight (nt, 0.89) and loose (nl, 0.49) interpretation
- at, al associative concept expanded query, tight (at, 0.89) and loose (al, 0.49).

In the basic synonym queries the concepts of the original search requests were identified in the thesaurus and matching models for their synonymous expressions exceeding the indicated strength and reliability were used in query construction. No conceptual expansion was performed. In the narrower concept expanded queries, narrower concepts were first added and then the matching models for their synonymous expressions were used as above in query construction. In the associative concept expanded queries, both narrower and associative concepts were added and then corresponding matching models were collected as above.

Each request was formulated into conjunctive queries (operator AND), proximity queries (operator 'and.p' in TRIP and '#20' in INQUERY) and probabilistic sum queries (#sum in INQUERY) according to the expansion types. The average number of concepts in the requests was 4.2, which gave 3.8 *conjunctive* facets for conjunctive queries. In the case of proximity queries the average number of conjunctive facets was reduced to 2.6. The average number of keywords and phrases in different query types for the conjunctive and probabilistic sum queries in the INQUERY-environment were as follows: os: 4.2; bt: 7.6; bl: 12.2; nt: 15.9; nl: 21.6; at: 19.6; al: 40.2. In the TRIP-environment the figures were about the same. In proximity queries, corresponding figures are smaller due to a smaller number of concepts.

Statistical significance of the findings concerning the original and expanded queries were first tested by the Friedman two-way analysis of variance by ranks in each case of conjunctive, proximity and sum queries. In all cases the test indicated statistically significant ($p < 0.001$) differences for recall and precision among the expansion results. Next the pairwise statistical significance between the original query and each expansion was tested (Siegel & Castellan, 1989). The precision tests were two-tailed, and recall tests two-tailed except for the conjunctive and proximity queries where recall was bound to improve.

We first report expansion results for Boolean queries, see Figs. 8-9. The two retrieval systems behave differently due to differences in the index type (basic vs. inflected words) and operators (#20 vs. and.p). However, some trends are clear. Tight query interpretation yields higher precision and lower recall than the loose interpretation. Also expansions from basic synonym query to narrower concept query and associative concept query increase recall at the cost of precision in a rather systematic way. Performance improves in TRIP in tight expansion w.r.t. the original query. This is due to index type and the thesaurus yielding many precise compound words which are split in the index for INQUERY.

Table 1 gives statistical significance of the findings. For TRIP, the precision findings were not significant and are thus not given.

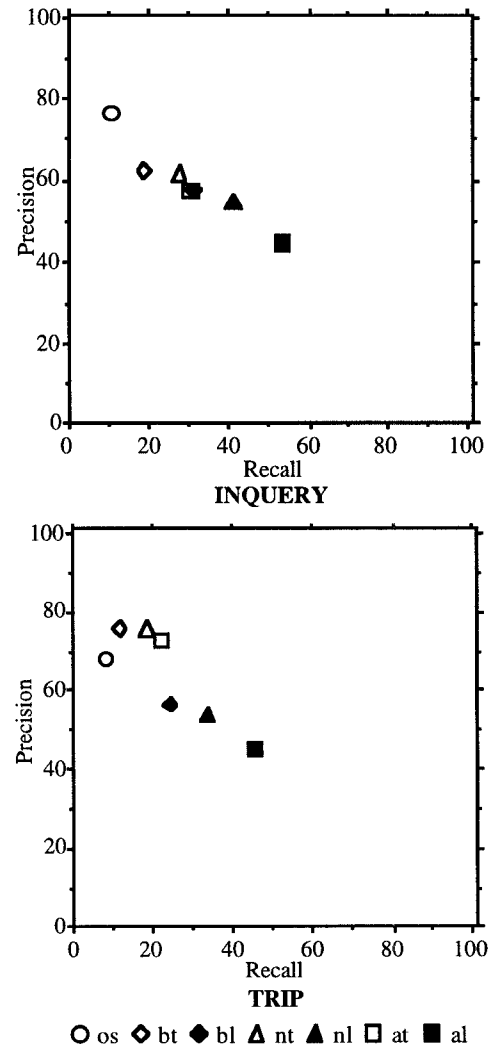

Fig. 8. Scattergram of exhaustive conjunctive queries with INQUERY and TRIP (N=10).

The expansion test results for the probabilistic sum queries for the 10 requests are given in Fig. 10. Each curve represents the average performance of one expansion type. The plotted points are recall - precision averages at the given retrieved set sizes from one to 100 documents for the expansion type in question (see Hull, 1993). It can be seen that all expansion types perform (on the average) better than the original query whereas the differences between the expansion types are minor. The curves end around 60% recall because of a large recall base. All pairwise differences in recall and in precision between the original query and each expanded query are statistically significant (p <0.001).

4. DISCUSSION AND CONCLUSIONS

We have presented a deductive data model for QE because the ordinary RDM cannot represent and manage essential thesaurus relationships in a natural way. We also described a QE software tool, called the ExpansionTool, for parametrized

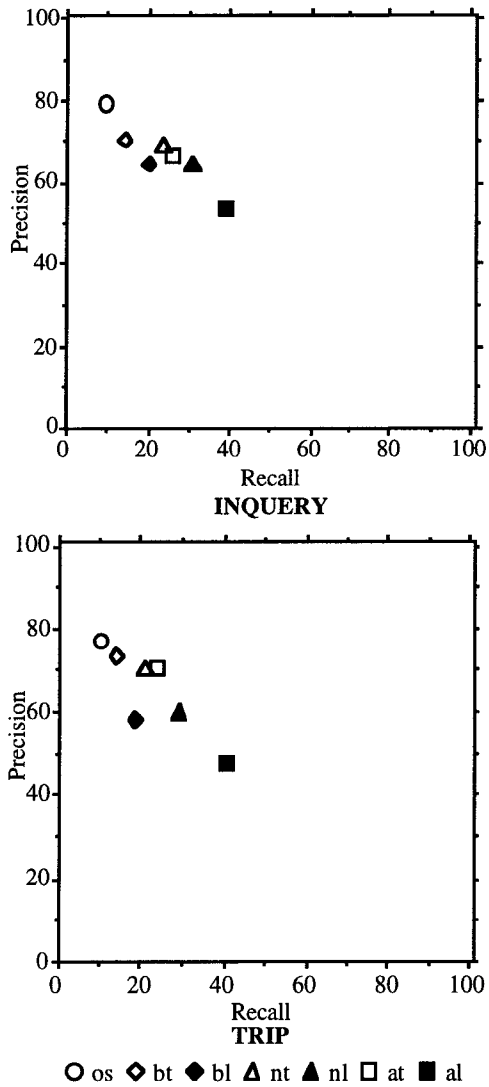

Fig. 9. Scattergram of proximity queries with INQUERY and TRIP (N=10).

Table 1. Statistical significance of the findings

Conjunctive queries		
INQUERY		**TRIP**
Recall	**Precision**	**Recall**
os - bl *	-	os - bl **
os - nt *	-	os - nt *
os - nl ***	-	os - nl ***
os - at **	-	os - at **
os - al ***	os - al **	os - al ***
Proximity queries		
INQUERY		**TRIP**
Recall	**Precision**	**Recall**
os - nl **	-	os - nl **
os - at *	-	os - at *
os - al **	os - al **	os - al ***

* p <0.05, ** p <0.01, *** p <0.001

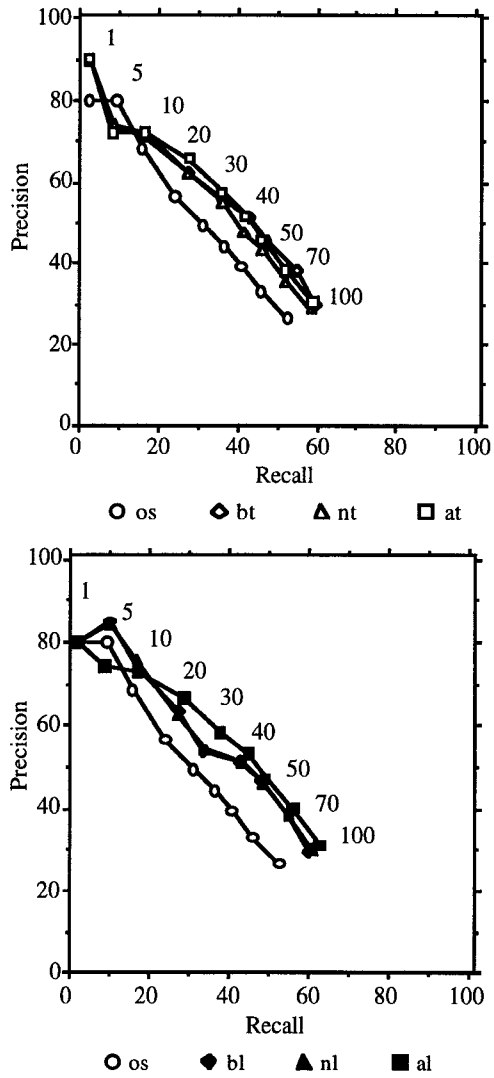

Fig. 10. Line chart of probabilistic sum queries with INQUERY (N=10).

automatic QE intended (i) for use prior to the initial search, (ii) for natural language retrieval of document lacking intellectual indexing, (iii) in heterogeneous retrieval environments where the database index types, retrieval systems and matching methods vary. The tool utilizes a searching thesaurus represented in the data model.

The expansion examples show that the ExpansionTool makes it easy to generate a range of quite differently behaving queries to a number of search environments which are heterogeneous with respect to the overall retrieval strategy, query language properties and database index construction strategies. The findings of the retrieval tests suggest that, through the ExpansionTool approach, (1) it is possible to increase recall in the Boolean environment systematically at a negligible cost in precision by applying the expansion parameters, and (2) retrieval performance improves in a ranked retrieval system (cf. the negative findings by Jones & al., 1995).

We considered query formulation based on user's identification of the concepts relevant to her/his needs. In order to support the identification of relevant concepts, the relations can be augmented by a relation **WORDS(WORD,**

ENO, COMPTYPE) giving for each word all expressions where it is a component of type **COMPTYPE**, e.g., a compound word component (cf. Jones, 1993). Then information needs can be matched with the thesaurus through **WORDS** using techniques suggested by Jones (1993).

We also tested the weighted sum operator, with the same query structure as for the sum operator but varying weights giving less weight to all keys except for the key corresponding to the most reliable matching model of each concept. However, the results were never better than those obtained with the presented unweighted query structure. Still, we are working toward supporting the probabilistic weighted sum operator, with automatic weight computation based on thesaurus relationships. The deductive operations will also be extended for cyclic transitive relationships.

Thesaurus construction would benefit from tools for acquiring domain knowledge. The knowledge represented in the model may be customized to each particular user to support his/her cognitive structures. Thus domain knowledge should be collected from the users interactively (Paice, 1991; Das & Croft, 1990). The matching models should be generated automatically from expressions. This requires integration of NLP-tools. The useability of ordinary (indexing) thesauri in the construction of a thesaurus for automatic QE presents a problem due to defective hierarchies. If related concepts or partitive narrower concepts are put into a hierarchy of generic narrower concepts, expansion will not work properly. This is, however, common in ordinary (indexing) thesauri, originally intended for human use.

The data model and the integrated query language are also usable as a tool for managing an indexing thesaurus (Järvelin & al., 1996). The main application area of the ExpansionTool, however, is filter agents for networked information retrieval. Obviously, the ExpansionTool approach can be utilized in improving the parametrizability and matching expressions of information filter agents of networked heterogeneous database environments.

Acknowledgment: The INQUERY retrieval system was used in part of this research. The INQUERY software was provided by the Information Retrieval Laboratory, University of Massachusetts Computer Science Department, Amherst, MA, USA.

REFERENCES

Abramson, H. & Dahl, V. (1989). *Logic Grammars.* Heidelberg: Springer-Verlag.

Agrawal, R. (1987). Alpha: An extension of relational algebra to express a class of recursive queries. In: *Proceedings of the 3rd International Conference on Data Engineering.* Washington, D.C.: IEEE Computer Society Press; 1987, pp. 580-590.

Chang, C.L. & Walker, A. (1986). A Prolog programming interface with SQL/DS. In Kerschberg L. (ed.), *Expert Database Systems: Proceedings from the 1st International Workshop.* Menlo Park, CA: Benjamin-Cummings, pp. 233-246.

Croft, W.B. (1986). User-Specified Domain Knowledge for Document Retrieval. In: Rabitti, F. (ed.) *Proc. of the 13th International Conference on Research and Development in Information Retrieval* Pisa, Italy.

Croft, W. B. & Das, Raj (1990). Experiments with query acquisition and use in document retrieval systems. Vidick, J-L. (ed.) *Proc. of the 13th International Conference on Research and Development in Information Retrieval.* Bruxelles: ACM, pp. 349-368.

Ekmekcioglu, F.C., Robertson, A.M. &Willett, P. (1992). Effectivenes of query expansion in ranked-output retrieval systems. *Journal of Information Science, 18,* 139-147.

Fidel, R. & Efthimiadis, E.N. (1995). Terminological knowledge structure for intermediary expert systems. *Infomation Processing and Management, 31*(1), 15-27.

Hancock-Beaulieu, M. (1992). Query expansion: Advances in research in online catalogs. *Journal of Information Science, 18,* 99-103.

Hull, D. (1993). Using statistical testing in the evaluation of retrieval experiments. Korfhage, R., Rasmussen, E.M. & Willett, P. (ed.) *Proc. of the 16th International Conference on Research and Development in Information Retrieval.* New York, NY: ACM, pp. 349-338.

ISO (1986). *ISO International Standard 2788. Documentation - Guidelines for the establishment and development of monolingual thesauri.* International Organization for Standardization.

ISO (1993). *ISO International Standard 8777:1993(E). Information and documentation — commands for interactive text searching.* International Organization for Standardization.

Jones, S. (1993). A thesaurus data model for an intelligent retrieval system. *Journal of Information Science, 19,* 167-178.

Jones, S. & al. (1995). Interactive thesaurus navigation: intelligence rules OK? *Journal of the American Society of Information Science, 46*(1), 389-404.

Järvelin, K. & Kristensen, J. & Niemi, T. & Sormunen, E. & Keskustalo, H. (1996). A deductive data model for thesaurus navigation and query expansion. Tampere, Finland: University of Tampere, Dept. of Information Studies, Report RN-1996-1, to be published.

Koskenniemi, K. (1983). Two-level morphology: a general computational model for word-form recognition and production. Helsinki: University of Helsinki, Dept. of General Linguistics, Publication No. 11.

Kristensen, J. (1993). Expanding end-user's query statements for free text searching with a search-aid thesaurus. *Information Processing & Management, 29*(6), 733-745.

Niemi, T. & Järvelin, K. (1992). Operation-oriented query language approach for recursive queries – Part 1. Functional definition. *Information Systems, 17 (1),* 49-75.

Paice, C.D. (1991). A thesaural model of information retrieval. *Information Processing & Management, 27*(5), 433-447.

Pereira, F.C.N. & Warren, D.H.D. (1980). Definite Clause Grammars for language analysis — A survey of the formalism and a comparison with Augmented Transition Networks. *Artificial Intelligence, 13*: 231-278.

Siegel, S., & Castellan, J. (1989). Nonparametric statistics for the behavioral sciences. New York: McGraw-Hill.

Ullman, J.D. (1989). *Principles of Database and Knowledge Base Systems. Vol. II: The new technologies.* Rockville, MD: Computer Science Press.

UMLS (1994). *UMLS Knowledge Sources.* 5th Experimental Edition. Bethesda MD: National Library of Medicine.