CIRI - An Ontology-based Query Interface for Text Retrieval

Eija Airio, Kalervo Järvelin, Pirkko Saatsi,
Jaana Kekäläinen, and Sari Suomela

Department of Information Studies
University of Tampere, Finland
{eija.airio, kalervo.jarvelin, pirkko.saatsi, jaana.kekalainen, sari.suomela}@uta.fi

Abstract. Ontologies can be used in IR for document indexing, query formulation and automatic query expansion. The three-level model developed in UTA allows expressing essential ontology relationships for IR. The levels of the model are the conceptual level, the linguistic level and the occurrence level. The first of the levels includes hierarchically structured concepts, the second expressions corresponding the concepts, and the last string models corresponding the expressions. The CIRI system (Concept-based Information Retrieval Interface) is based on the three level model. In the CIRI interface, the user is able to open available ontologies and select concepts from them, select available search engines and databases, select the query expansion level, create a query from selected concepts and submit the query to the search engine.

1 Introduction

Thesaurus databases have been widely used in IR. It is possible to use a thesaurus database interactively for document indexing and query formulation. Another way to use it is automatic query expansion [1].

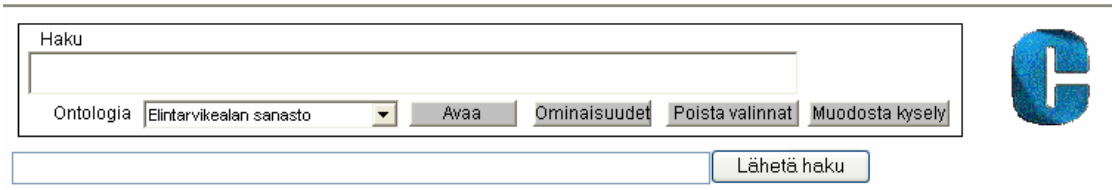
Järvelin and colleagues have presented a deductive data model for thesauri in order to manage essential thesaurus relationships in a natural way in IR and query expansion. The model is based on the idea of the three abstraction levels: **conceptual level**, **linguistic level** and **occurrence level**. The conceptual level contains hierarchically structured **concepts**. The type of the **hierarchical relationship** (for example *generic*, *partitive* or *instance*) is included in the model as well. The model is based on the supposition that appropriate relationships are defined for each ontology separately. In addition of hierarchical relationships, **associative relationships** between concepts may be defined. The linguistic level contains **expressions** corresponding concepts. There may be none, one, or many expressions for each concept. The expressions may be expressed in various languages, both natural and artificial. The expressions referring to a concept may be interpreted to be synonyms with each other. Each expression has one or more counterparts on the **occurrence** level, which includes matching models corresponding expressions. Occurrences may be for example basic words, compound words or stemmed words, depending on target index types [1].

The word **ontology** is often used nowadays instead of thesaurus. An ontology describes concepts and entities of an application environment, as well as its terminology. An ontology has to be precise and explicit, because it must be computer-readable as well [2].

2 The CIRI system

The three-level model is applied in the CIRI system (Concept-based Information Retrieval Interface) developed in the University of Tampere. The CIRI system is implemented with Java servlets, and is used via a web browser. In the interface, the user is able to open available ontologies and select concepts from them, select available search engines and databases, create a query from selected concepts and submit the query to the search engine (see Figure 1).

In the CIRI interface, the user can select the ontology from the menu, and open it by pressing the *open* (*avaa*) button. In the ontology view, it is possible to browse the ontology by opening and closing branches (see Figure 2). When the user has selected all concepts she wants to be included into the query, she can press the *create query* (*muodosta kysely*) button. After that the user presses the *submit the query* (*lähetä haku*) button. The retrieval result is shown under the CIRI applet (see Figure 3). The user may see whole documents by clicking titles, and return to the document list.



Ciri-haku

Käyttöohjeet

1. Paina Ominaisuudet -nappia ja valitse haluamasi tietokanta.

2. Valitse sanasto kohdan *Ontologia* alasvetovalikosta ja paina Avaa. Sanaston navigointipuu aukeaa omaan ikkunaansa.

3. Valitse sanastosta haluamasi käsitteet, sekä montako tasoa haluat käsitteitä laajentaa. Sitä mukaa, kun valitset käsitteitä, hakulause muodostuu kohtaan *Haku*.

4. Kun olet valinnut kaikki haluamasi käsitteet, älä sulje Ontologia-ikkunaa, vaan jätä se auki. Paina *Muodosta kysely*, jolloin Ciri laajentaa valitut käsitteet haluttuun tasoon asti, ja muuttaa laajennuksen tulokset niitä vastaaviksi täsmäytysmalleiksi. Tämä laajennettu kysely ilmestyy haku-appletin alapuolella sijaitsevaan tekstikenttään. Voit vielä halutessasi muokata syntynyttä hakua tai lisätä sen perään omia hakusanojasi. Suorita haku painamlla *Lähetä haku*. *Poista valinnat* -nappi tyhjentää kaikki valinnat.

Fig. 1. The CIRI interface

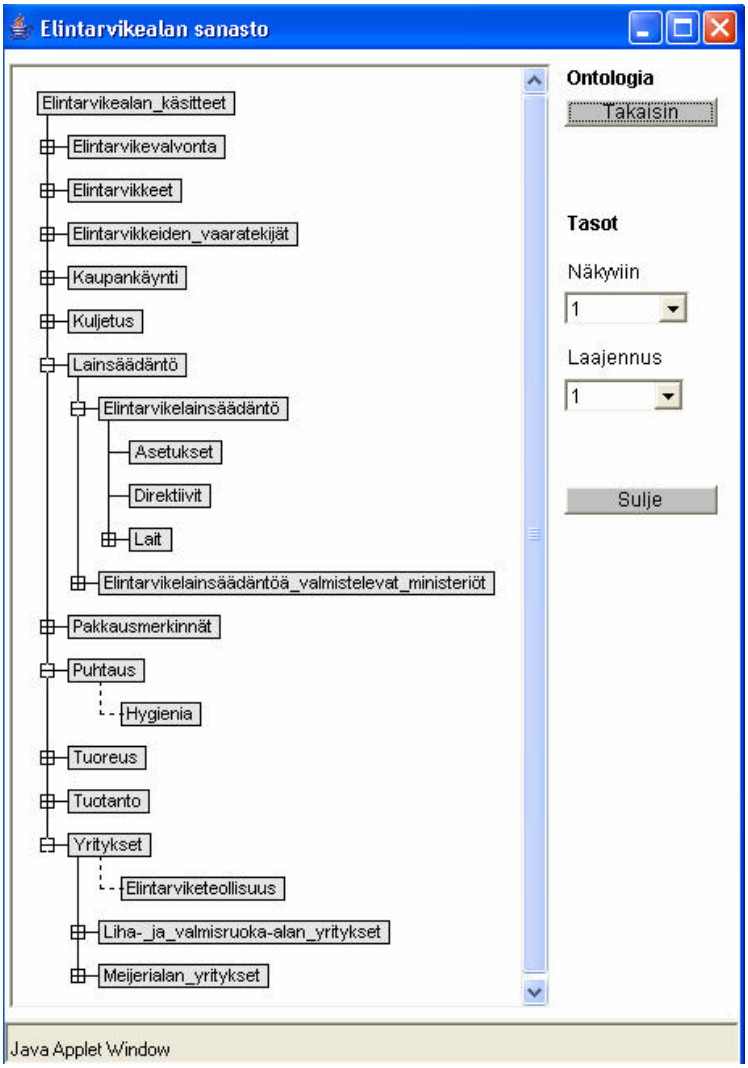


Fig. 2. The ontology view of CIRI

The dash line in the ontology stands for an association relation. In the *visible* (*näkyviin*) menu, the user may select the number of shown levels. The *expandable* (*laajennus*) menu gives the possibility to choose how many levels of narrower concepts will be taken with the query. This does not concern associative relations: only one level of them is chosen. When the user clicks the mouse on a concept, it is chosen to the query, and it can be seen in the *query* (*haku*) field in the interface. Removing selected concepts from the query is possible by re-clicking the concept.

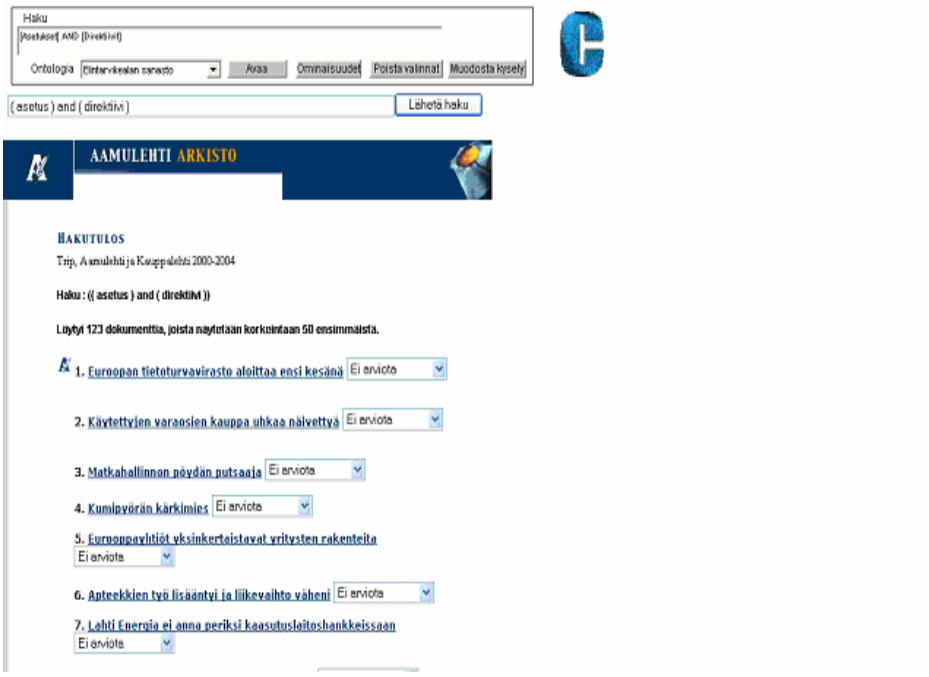


Fig. 3. The retrieval result

When the user presses *open the ontology* (*avaa*) button, a servlet retrieving concepts and concept relations in the sql-database is started (see Figure 4). These are redirected to a servlet which depicts a graphical presentation of the ontology. When the user presses the *create query* (*muodosta kysely*) button, a servlet retrieves expandable concepts, as well as corresponding expressions and occurrences in the sql-database, based on information given by the user (selected concepts and expansion level). Retrieved information is redirected to a servlet which formulates the final query. This servlet needs also user information about the search engine type, because the type affects the query syntax.

A servlet submits the query to the search engine, using information given by the user (the search engine and the database). Finally, retrieval result is submitted to the CIRI interface.

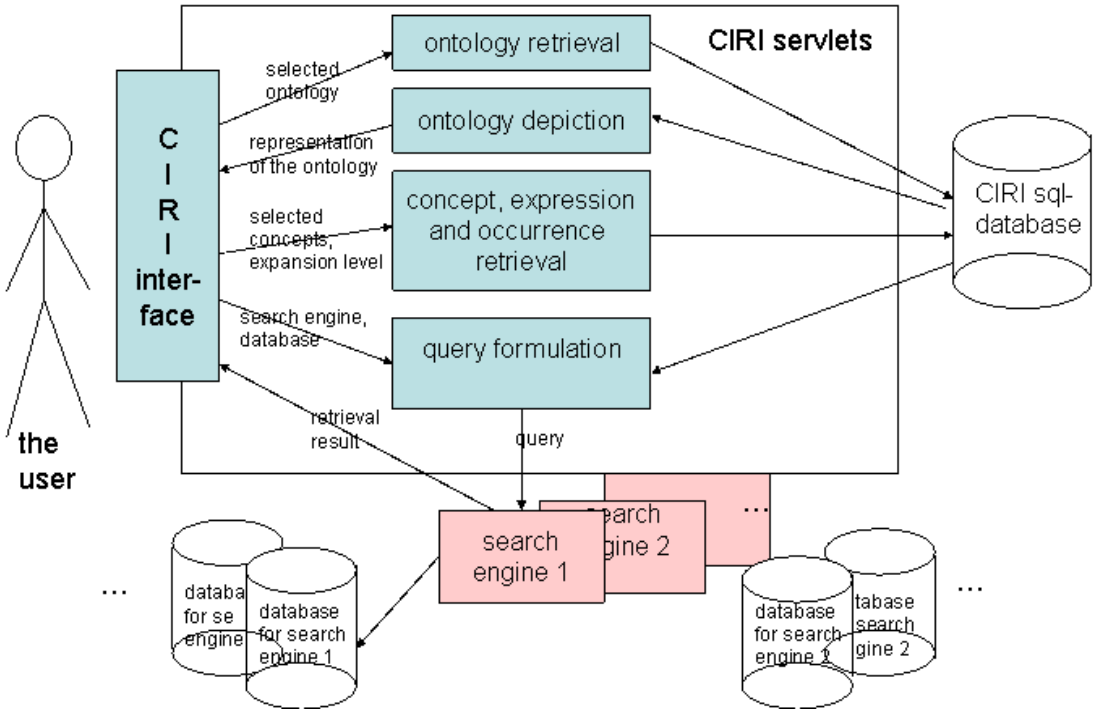


Fig. 4. The functionality of CIRI

3 Implementation of the three-level model

An SQL-database is a natural solution to be applied in this type of task, where we have a lot of relations between various items, and some of the relations are bidirectional. We have selected the PostgreSQL open source database system[1] to store the concepts, expressions and occurrences, as well as relationships between them.

The structure of the CIRI database is given in Figure 5. The items of all the present ontologies are stored in the same CIRI database. Each table includes an attribute ontology_id, which is used as an identifier between ontologies.

The table **concept** is used to store attributes of the concepts. For each concept is given a unique identification number, which is stored in a field **cno**. The field **cname** includes the name of the concept, while the field **category** refers to the type of the

[1] http://www.postgresql.com

concept (for example *person*, *process* or *economy*). The field category includes codes referring to id:s in the table ctype. The table **ctype** includes all the possible concept categories for concepts, as well as unique codes for them. In the field **definition** a short description of the concept may be stored.

Relations between the concepts are described in the table **concept_relation**. Each relation is given a unique identification number **rno**. Fields **broader_cons** and **narrower_cons** include the identification numbers of the referred concepts. The field **strength** includes the strength of the relation between the concepts. It is possible to adjust the CIRI system so that expansion is done only if the strength between the concepts is high enough. The field **rel_type** includes the identification number of the relation type (for example *whole-part*, *generic*, *associative*). Association relations are always bidirectional, and they are therefore stored twice in the concept relation table. Attributes of the relation corresponding the relation type number are stored in the table **relation_type**.

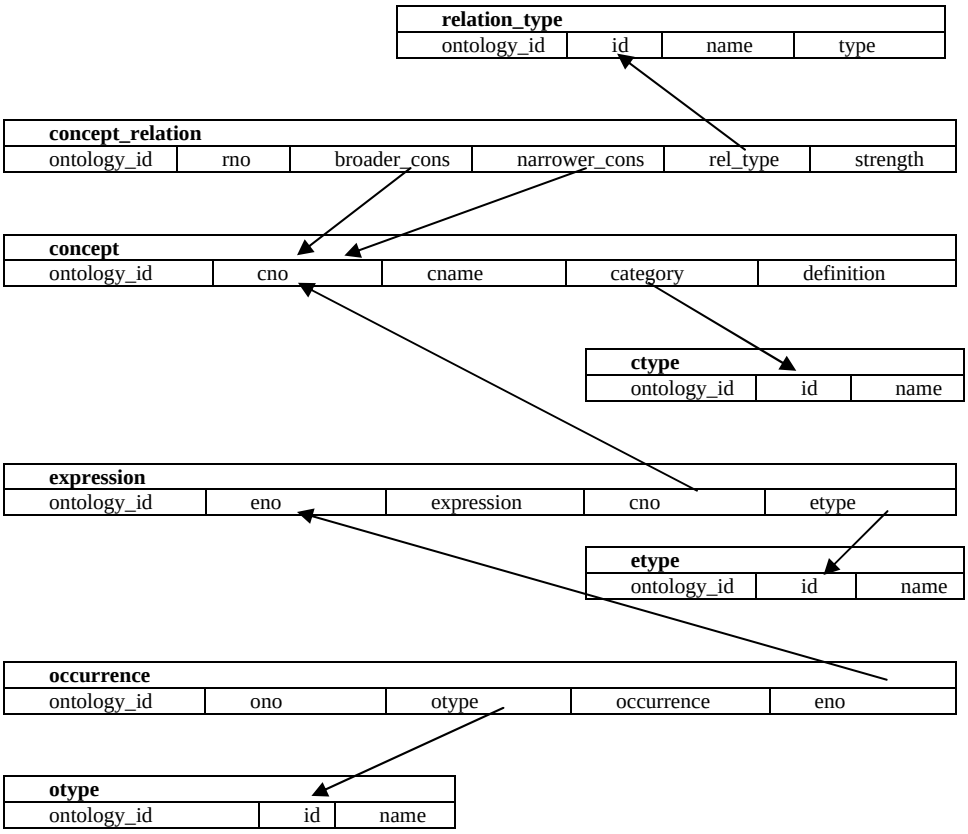


Fig. 5. Representation of the tables in the CIRI database

Expressions are stored in the table **expression**. Each expression has a unique id number **eno**. The name of the expression is stored in the field **expression**. The field

cno includes the identification number of the corresponding concept. The field **etype** includes the id number of the type of the expression. The names of the expression types (for example *term*, *verb*, *synonym*) are stored in the table **etype**.

The string attributes corresponding the expressions are stored in the table **occurrence**. Each occurrence has a unique id number **ono**. The field **otype** includes an id number referring to the table **otype**, which includes names of the occurrence types. The field **occurrence** includes the string, and the field **eno** includes the id number.

When the user opens an ontology, CIRI reads the concept and concept_relation tables, and creates the ontology view based upon this information. In query creation, table concept_relation is used to find possible expandable concepts. The assumed level of expansion is three, unless the user changes it. Expansion is done from broader concept to narrower concepts. In associative relations, only one level is expanded. The expressions corresponding the concepts are selected from the table expression, and finally occurrences corresponding expressions are taken with the query. CIRI creates the query from the occurrences based on the search engine information given by the user (because search engines have different query syntax).

4 Creating three-level ontologies with the Protégé ontology editor

We have used the Protégé ontology editor for ontology creation, because it is a widely used open source ontology editor, and used by many Fenix project participants as well. Protégé is an open source Java-tool developed by Stanford Medical Informatics at the Stanford University School of Medicine[2]. Protégé gives the possibility to separately define **classes** of information (schema) and store **instances** of these classes. By utilizing **slots** in Protégé it is possible to define features of classes or relationships between classes, as well as properties of instances [3]. Classes of Protégé correspond concepts in the three-level model, while instances correspond expressions.

It is possible to import and export files in many formats in Protégé in addition to its internal format: RDF, XML and OWL, utilizing plugins for Protégé. **OWL format** is utilized within this research. In OWL format, Protégé applies the term **property** is instead of the term slot, and the term **individual** instead of the term instance.

Protégé does not have the ability to deal with the three-level model. Protégé is able to store concepts hierarchically, but it does not give a possibility to define the types of the hierarchical relationships. Protégé does not support associative relationships between concepts at all. The second level, expression level, is possible to introduce in Protégé: one or more expressions corresponding a concept may be created. Protégé does not support introducing the third level, the occurrence level, at all.

We have tried to bypass the deficiencies of Protégé for introducing the desired features of the three-level model. To express the type of the concept relationships, we have created a **property** for each relationship type (for example properties *associative* and *whole-part*). These properties are used to attach a concept with another. The property **strength** is attached with the relationship when it is created to define

[2] http://protege.stanford.edu/index.html
[3] http://protege.stanford.edu/doc/users_guide/index.html

numerically the strength of the relationship. The problem of omitted occurrence level is solved by utilizing properties as well. For each expression, 1-6 property triples[4] are included: occurrence string and occurrence type.

The three-level ontologies created with Protégé may be used for various purposes. When the occurrence level is not needed, it may be omitted and the concept level with relation types, as well as the expression level may be utilized. (It is easy to remove the properties not needed by a simple script from the OWL-file.) When an ontology created with Protégé is used in CIRI, the whole three-level structure will be needed. We have compiled a php-script to handle the OWL-file created by Protégé for this purpose. The script takes the OWL-file and the ontology_id of the new ontology as input, and creates the insert-commands to insert all desired items into the PostgreSQL-database of CIRI. The script goes through the OWL-file several times, searching information of desired insertable items.

5 Ontology creation as an intellectual task

Creating the ontology is not an easy task: the creator has to familiarize with the context of the ontology, and possibly define the users of the forthcoming ontology. Appropriate literature: textbooks, reference books, encyclopedias as well as newspapers, may serve as information sources, when creating the **concept hierarchy**. If there are prior ontologies or thesauri, classification schemes or glossaries dealing with the subject or some sector of the subject, they are worth as well. For creating sub-concepts, translation dictionaries may be useful: often they give, in addition to translations, synonyms or almost-synonyms of the input word in source language, which may serve as sub-concepts. Advices of possible field experts are valuable as well. Discussions with possible forthcoming users are vital in the beginning phase of ontology creation. Users may be asked to write down which they think are the most important concepts of the area, or they may be asked to create a mind-map of the subject.

The **expression level** contains none, one or more expressions corresponding each concept. Linguistic tools for synonym creating would be appropriate for this phase. Unfortunately, synonym lexicons often are too general for this purpose. Translation dictionaries, especially special dictionaries, giving synonyms in source language may be useful, as they are in concept creation phase also. If the ontology is going to be multilingual, translation dictionaries are vital in order to create expressions in desired languages. Usually, the expressions of the ontology are nouns, but sometimes it is reasonable to add related words of other word classes (verbs or adjectives) as well. For example, if we have a concept *Inspector* in the ontology, we may insert an expression *inspect* in addition of *an inspector*.

The **occurrence level** contains strings corresponding each expression. The purpose of the occurrences is to help information retrieval. The type of the occurrence is stored in the occurrence table of CIRI. The occurrence type references to certain type of index (or certain way to use the index). Possible occurrence types are the

[4] It is possible to create more property triples when desired

lemmatized occurrences, **stemmed** occurrences, **truncated** occurrences and **inflected** occurrences. NLP (natural language processing) tools are useful in occurrence creation. When possible, it is recommendable to utilize the same NLP tools in occurrence creation, which were used in database indexing. The most comfortable way for the user would be to integrate the NLP tools with the Protégé ontology editor: when the user selects the type of the occurrence, the plugin inserted in Protégé starts appropriate NLP tool, which offers suitable occurrence variants for the user.

A **lemmatizer** is useful when creating basic form occurrences, especially in the case of compound splitting, and a **stemmer** for creating stem occurrences. These are the simplest alternatives, both to implement and to use. When we are creating occurrences for an inflected index, we must know the type of the IR system. There are two possibilities: in a system based on a **probabilistic model**, truncation is not possible, while in a system based on a **Boolean model** truncation can be used. In the latter case, a **stem generator** is useful: just the truncation character has to be added to the strings given by the stem generator. When truncation is not possible, the user should provide all the inflected word forms when creating occurrences. That is possible for a language which does not have many cases (for example English with two cases), but is harder with a highly inflected language (for example Finnish with 14 cases). A tool generating word cases would be convenient for this purpose, but tools like that may not be available. The following approach might be useful to overcome this shortcoming: first a stem generator is used to create word beginnings, after which target index is grepped with these strings (see [3]). Manual inspecting or automatic checking is vital in this approach, because grepping gives presumably much garbage. For example a Finnish stem generator (see [3]: minstemma) gives following strings for the word *kissa* (a cat): *kissa, kisso*. Grepping a Finnish index with these strings gives a result including 69 strings. Of these only 18 are inflected variants of the word *kissa*. To remove garbage, a lemmatizer should be utilized to check the grepping result (this naturally applies only for the words recognized by the lemmatizer).

6 Conclusion

We have introduced CIRI, an interface for concept-based information retrieval. In CIRI, the representation of concepts is based on three abstraction level model: concepts, expressions naming the concept, and occurrences representing the expressions in queries. Concepts are represented to users as tree hierarchies, which they can browse to get an idea of the conceptual relations in given subject area, and from which they may choose the concepts for their queries. When a retrieval system is selected a query is formulated automatically from the chosen concepts. The user does not have to seek for the alternative names for concepts, nor be acquainted with the syntax of query languages. At the time of writing this paper we are starting tests with CIRI to find out how users will receive it in realistic settings.

References

1. Järvelin, K., Kristensen, J., Niemi, T., Sormunen, E., Keskustalo, H.: A Deductive Data Model for Thesaurus Navigation and Query Expansion. Finnish Information Studies 5. University of Tampere, Finland (1996)
2. Hyvönen, E.: Ontologiat: standardeja, työkaluja, sovelluksia, tutkimusta, http://www.cs.helsinki.fi/u/eahyvone/presentations/1%20OntologyPerspectives.pdf (2003)
3. Kettunen, K., Kunttu, T., Järvelin, K.: How to tackle the morphological variation of Finnish in a probabilistic best match IR environment? [Submitted] (2004)