February 2005

On the Query Refinement in Searching a Bibliographic Database

Nenad Stojanovic
University of Karlsruhe, Germany

In: Ferstl, Otto K, u.a. (Hg) 2005. *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety*; 7. Internationale Tagung Wirtschaftsinformatik 2005. Heidelberg: Physica-Verlag

ISBN: 3-7908-1574-8

On the Query Refinement in Searching a Bibliographic Database

Nenad Stojanovic
University of Karlsruhe, Germany

Abstract: In this paper we present an application of the logic-based query refinement in the searching for information in an information portal. The refinement approach is based on the discovery of causal relationships between queries regarding the inclusion relation between the answers of these queries. We define a formal model for the query-answering pairs and use methods from the inductive logic programming for the efficient calculation of a (lattice) order between them. In a case study we demonstrate the benefits of using our approach in the traditional information retrieval tasks. We focus on the combination of the free-text based querying and the logic-based query refinement.

Keywords: Query Refinement, Ontology, Bibliographic Database

1 Introduction

The growing nature of the (public available) information implies a users behaviour's pattern that should be treated in a more collaborative way in the modern retrieval systems: users tend to make short queries which they refine subsequently. Indeed, in order to be sure to get any answer to a query, a user forms as short as possible query and depending on the list of answers, he tries to narrow his query in several refinement steps. The main problem is that a user cannot express his information need straightforwardly in a query posted to an information repository, i.e. a user's query represents just an approximation of his information need [Sar75]. Consequently, a query should be refined in order to ensure the retrieval of as much as relevant information. Unfortunately, most of the retrieval systems do not provide a cooperative support in the query refinement process, so that a user is "forced" to change his query on his own in order to find the most suitable results. Indeed, although in an interactive query refinement process [Eft95] a user is provided with a list of terms that appear frequently in retrieved documents, the explanation of their impact on the retrieval process is completely missing. Consequently, some redundant and/or failing refinements can be suggested to a user, what decreases the efficiency of the refinement process drastically.

In our previous work we developed a logic-based approach for refining queries that is based on the usage of an ontology for modelling an information repository [Sto04]. The approach enables a user to navigate through the information content incrementally and interactively. In each refinement step a user is provided with a complete but minimal set of refinements, which enables him to develop/express his information need in a step-by-step fashion.

In this paper we apply this approach for searching a traditional bibliographic database. Since the data is structured according to a schema defined by a database provider, we first migrated this schema into an ontology. By using this ontology the content of the database is translated into a knowledge base. Each user's query is mapped into an ontology-based query and the logic-based query refinement is performed. The refinements are ranked according to their informativeness and displayed to the user.

This approach supports the so called step-by-step query refinement, which enables a novel user to inspect the content of the bibliographic database in a more systematic manner. Our evaluation study shows two main advantages of such a refinement: (i) a user can find relevant documents faster and (ii) he is more satisfied with the relevance of the documents for his information need.

The paper is structured in the following manner: In Section 2 we present the basic terminology we use in this paper as well as the logic-based query refinement approach. Section 3 outlines the problems that could arise in an information retrieval process. In Section 4 we present a bibliographic case study that illustrates how our query refinement approach can be used in resolving these problems. Section 5 contains discussion about related work. In Section 6 we give concluding remarks.

2 Background

In this section we give the basic assumption/terminology we use in this paper. Moreover, we sketch the logic-based query refinement process we use latter in the paper. More details can be found in [Sto04].

Definition 1: Ontology

An ontology is a structure $O := (C, \leq_C, R, \sigma)$ consisting of:

- two disjoint sets *C* and *R* whose elements are called concept identifiers and relation identifiers, resp.,
- a partial order $\leq_C$ on C, called concept hierarchy or taxonomy (without cycles)
- a function $\sigma : R \rightarrow C^+$, called signature

Often we call concept identifiers and relation identifiers concepts and relations, respectively, for the sake of simplicity.

Definition 2: Domain and Range

For a relation $r \in R$ with $|\sigma(r)| = 2$, we define its domain and its range by $dom(r) := \pi_1(\sigma(r))$ and $range(r) := \pi_2(\sigma(r))$.

Definition 3: Knowledge Base

A Knowledge Base is a structure $KB := (C_{KB}, R_{KB}, I, l_c, l_r)$ consisting of:

- two disjoint sets C_{KB} and R_{KB}
- a set I whose elements are called instance identifiers (or instances or objects shortly)
- a function $l_C : C_{KB} \rightarrow I$ called concept instantiation
- a function $l_r : R_{KB} \rightarrow I^+$ called relation instantiation

A relation instance can be depicted as $r(I_1, I_2, ..., I_n)$, where $r \in R_{KB}, I_i \in I$. Similarly, $c(I_i)$, where $c \in C_{KB}$, represents the concept the instance I_i belongs to. r is called a predicate and I_i is called a term. Note that in this work we treat only binary relations. However the extension for n-ary relations is straightforward.

Definition 4: Query

A conjunctive query is of the form or can be rewritten into the form: $Q(\overline{X}) \equiv forall\ \overline{X}\ \overline{P}(\overline{X}, \overline{k})$, with $\overline{X}$ being a vector of variables $(X_1, ..., X_n)$, $\overline{k}$ being a vector of constants (concept instances), $\overline{P}$ being a vector of conjoined predicates (relations).

For example, for the query "forall x,y worksIn(x, KM) and researchIn(x, y)"

we have $\overline{X} := (x, y)$, $k := (KM)$, $\overline{P} := (P_1, P_2)$, $P_1(a,b,c) :=$ `worksIn(a,b)`, $P_2(a,b,c) :=$ `researchIn(a,c)`.

Since a predicate constrains the interpretation of a variable in a query, in the rest of the text we will use the term *query constraint* as the description of a predicate. For example, `researchIn(x, y)` is a constraint for the interpretation of the variable x.

This is the standard form of queries considered in similar research [Cha90], [Go97]. Moreover, our limiting focus to conjunctive queries is not a serious limitation since the result of a disjunctive query can be considered as the union of the results of the disjuncts; that is, each disjunct can be considered as an independent query.

Definition 5: Answers (results) of a query

Let Σ be the set of all relation instances which can be proven in the given knowledge base.

For a query Q = " $forall\ \overline{X}\quad \overline{P}(\overline{X},\overline{k})$ " an answer is an element (tuple) in the set $R(Q) = \{\overline{X}\} = \{(x_1,x_2,...x_n)\}$, such that $\overline{P}(\overline{X},\overline{k})$ is provable, i.e. each of the relation instances $r(x_1,x_2,...,x_n,k_1,...k_l),\ r \in \overline{P}$ exists in the set Σ.

If a query cannot be proven (i.e. it returns zero results) it is called a *failing* query.

Definition 6: Extended query / Query refinement operator

An extended query is a pair <Q, A>, where A is a boolean function, called *Acceptance Test*, that takes as input the answer generated by executing the query Q over a knowledge base *KB* and returns {True, False}. We say that an extended query is acceptable for the *KB* iff $A(R(Q))$ = true. The concept *Acceptance Test* is introduced in [Cha90] as a possibility to express constraints in user's queries that cannot be expressed in the relational language, for example that the user is willing to accept answers from a modified query.

The goal of a *query refinement operator* is to transform a query <Q, A> to a query <Q', A> so that latter is acceptable. An extended query <Q', A> refines <Q, A> (in the notation <Q, A> $\rightarrow_{\text{ref}}$ <Q' A>, or as a shorthand[1] $Q \rightarrow_{\text{ref}} Q'$), if in the context of the given knowledge base *KB* and the ontology O holds $R(Q') \subseteq R(Q)$. It is clear that a refinement operator ($\rightarrow_{\text{ref}}$) derives a set of refinements for a query Q, or more formally:

$$Q \rightarrow_{\text{ref}} \{Q' \mid R(Q') \subseteq R(Q)\}.$$

Definition 7: Logic-based query refinement operator

From the model-theoretic point of view, the query refinement process can be treated as a logic implication, i.e. a query Q' implies another query Q (Q logically entails Q'). Since the ontology and the schema are the only constraints used for driving the refinement process, then

$$(Q \rightarrow_{\text{ref}} Q') \equiv (KB,O \vdash Q' \rightarrow Q),$$

where $\vdash$ depicts the derivation (inference) process.

However, due to the complexity of subsumption reasoning, we use an alternative, more tractable generality order, θ-subsumption, frequently used for efficient implementation of inductive logic programming tasks [DeR96].

The logic-based refinement of a query Q, denoted $\rho_l(Q)$ is a query obtained in one of the following ways:

---

[1] In the rest of text we will *implicitly* assume that each refinement is related to an *Acceptance Test*

(a) $\rho_l(Q) = Q\theta$, where θ is a substitution that corresponds to the variable x from Q;
(b) $\rho_l(Q) = (Q \wedge L)\theta^{-1}$, for some θ^{-1} (including $\theta = \{\}$), where θ^{-1} is some inverse substitution and L is a ground literal which is true in KB and there is a term t such that $t \in Q$ and $t \in L$

In both cases, (a) and (b):

1.- equ refinements, $R(\rho_{\text{logic-based}}(Q)) = R(Q)$, are aggregated to the initial query (i.e. they are treated as equivalent queries to the initial query) and

2.- non-minimal refinements, refinements which are subsumed by another (but not equ-) refinement A, $R(\rho_{\text{logic-based}}(Q)) \subset R(\text{A})$, are filtered.

An example:

In order to make the paper more readable, we give here a short example that illustrates the logic-based query refinement. Table 1 (left column) illustrates a small ontology. For example, a vehicle can be a car or a motorbike and can have several features. The right column in Table 1 represents a set of statements regarding our motivating example. For example, the car c5 is a `FamilyCar` and has a `GPS` system.

Ontology	**Knowledge Base**

`isA(Car, Vehicle)`
`isA(Motorbike, Vehicle)`
`isA(Luxury, Feature)`
`isA(Type, Feature)`
`hasLuxury(Luxury, Vehicle)`
`hasType(Type, Vehicle)`
`isA(Cabriolet, Luxury)`
`isA(Metallic, Luxury)`
`isA(Automatic, Luxury)`
`isA(GPS, Luxury)`
`isA(SportsCar, Type)`
`isA(FamilyCar, Type)`
	`Car(c1),hasType(SportsCar,c1), hasLuxury(Cabriolet,c1)`
	`Car(c2),hasType(SportsCar,c2), hasLuxury(Metallic,c2)`
	`Car(c3),hasType(SportsCar,c3), hasLuxury(Metallic,c3)`
	`Car(c4),hasType(SportsCar,c4), hasLuxury(Metallic,c4)`
	`Car(c5),hasType(FamilyCar,c5), hasLuxury(GPS,c5)`
	`Car(c6),hasType(FamilyCar,c6), hasLuxury(Automatic,c6)`
	`Car(c7),hasType(FamilyCar,c7), hasLuxury(Automatic,c7)`
	`Car(c8),hasType(FamilyCar,c8), hasLuxury(Automatic,c8)`
	`Car(c9),hasType(FamilyCar,c9), hasLuxury(Automatic,c9)`
	`Car(c10),hasType(FamilyCar,c10), hasLuxury(Metallic,c10)`
	`Car(c11),hasType(FamilyCar,c11), hasLuxury(Metallic,c11)`
	`Car(c12),hasType(FamilyCar,c12), hasLuxury(Metallic,c12)`
	`Car(c13),hasType(FamilyCar,c13), hasLuxury(Metallic,c13)`
	`Car(c14),hasLuxury(Automatic,c14)`

Table 1: A simple dataset used in motivating example

Figure 1 represent the results of applying logic-based refinement on the dataset. Note that our method enables the step-by-step refinement by defining in each step the minimal number of refinements. Therefore figure 1 presents two refinement

steps (e.g. in the first step only four refinements are generated – Automatic, <<FamilyCar, Metallic, SportsCar).

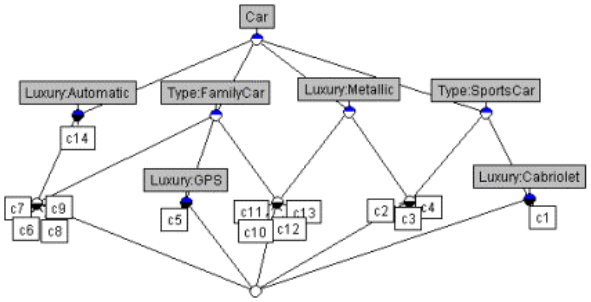

Figure 1: Lattice-based clustering of dependencies between product' features for the knowledge base depicted in Table 1. Note that a node encompasses all resources from its children nodes and all attributes from its parent nodes. E.g., the node {(`Luxury:GPS`), (`c5`)} contains the attribute (`Type:FamilyCar`) as well.

3 Information Retrieval Process

Information retrieval is usually considered as a querying process in which a user makes a query and tries to find information resources that are relevant for his information need. However, the process is more complicated since a query does not represent accurately a user's information need, but rather it is just an approximation of this need. Another problem is that users try to make as short as possible queries so that in the first querying step more relevant results are retrieved, which in subsequent query steps should be filtered. Therefore, an efficient retrieval system should support a *query refinement* process, in that a user is provided with enough information in order to make the right decision how to refine his query. For example an efficient grouping of results can be very useful for the user. Moreover, the system can ask a user some questions in order to acquire the user's preference and consequently to provide the best possible refinements. Figure 2 sketches the most important phase in an information retrieval process.

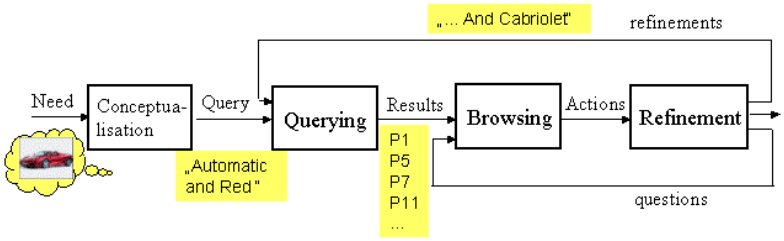

Figure 2: The Information Retrieval process

The Conceptualisation (cf. Figure 2.) is the process in which an, often ill-defined information need, is formalized in a set of query terms. The Querying process is usually implemented as a free-text search, that causes some degradation in the precision of the retrieval process (i.e. lots of irrelevant results can be retrieved). The Browsing process is usually well supported in traditional IR systems.

In an ontology-based information retrieval process, an ontology enhances the Querying process by performing an inference as a searching method. This ensures the maximal precision and recall of the retrieval process, i.e. all and only relevant results are retrieved. However, an ontology plays an important role in the disambiguation of a user's query [Sto03], helping in the better conceptualisation of a user's need. For example, if a user makes a very general query, e.g. Car(x) the ontology-based system can advice the user that his query is very ambiguous (since there are lots of cars) and that he should specify which type of the car he is interested in.

Finally, the in the Refinement process (cf. Figure 2.) an ontology enables the summarization of various refinements on the various levels of the granularity. This supports a step-by-step refinement of the user's query.

4 Logic-based Query Refinement in Searching a Bibliographic Database

4.1 The System

Recently, in order to enable more precise searching, traditional web information portals employ more semantics for the description of the information content. First, instead of a free-text query, some structuring of the content of a query is possible, e.g. according to the creation date and the author of an information resource. Second, the content of the information resources can be annotated using

terms from a predefined taxonomy, which enables the usage of a controlled vocabulary for generating more precise queries. However, although these modelling primitives can help in increasing the precision of the retrieval process, none of them is used for the refinement of users' queries. On the other hand, the well structured data and the hierarchical organisation of topics can be treated as a light-weight ontology. Therefore, our logic-based query refinement can be seen as an extension of the searching process in such portals. In this section we present an application of our query refinement approach in the domain of the bibliographic search, we have done in the scope of the SemIPort[2] project.

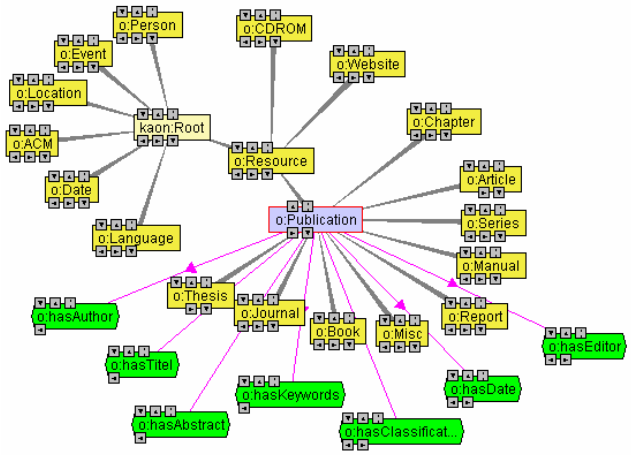

Figure 3: A part of the SemIPort ontology

CompuScience is a bibliographic database covering literature in the field of computer science, information- and communication technology, information management and science with about 160.000 citations. The database contains the complete computer science section from the Zentralblatt fuer Mathematik/ Mathematics Abstracts as well as all material from ACM's (Association for Computing Machinery) Computing Reviews and Guide to Computing Literature. The database aims at a complete coverage of European and American publications. Citations are in English and contain bibliographic information and indexing terms. Many records also include an abstract. The citations are classified according to the Computing Reviews Classification Scheme of ACM. Therefore, for a publication not only the "traditional" bibliographic data (like the author, the publication year) but also the metadata about its content w.r.t. ACM classification are given.

2 SemIPort (http://km.aifb.uni-karlsruhe.de/semiport/) is a Semantic Web related project, funded by the BMBF, whose task is the development of semantic methods for the traditional information portals.

Based on this data model we developed an ontology[3] for the bibliographic domain which is partially presented in Figure 3. As the ontology modelling language we use KAON (kaon.semanticweb.org). The XML version of the *CompuScience* dataset is converted in KAON *knowledge base*, whose part is presented in Figure 4. In that way we enable the integration of the presented query refinement approach into the retrieval process of the *CompuScience* search engine[4].

```
<a:Publication rdf:ID="CSall1175"
    a:hasTitle="An efficient randomized algorithm for detecting cir-
cles."
    a:hasYear="2001">
    <a:hasAbstract>Summary: Detecting circles from a digital image is
very important in shape recognition. In this paper, an efficient ran-
domized algorithm ….</a:hasAbstract>
    <a:hasAuthor rdf:resource="#CSall1176"/>
    <a:hasAuthor rdf:resource="#CSall1177"/>
    <a:hasClassification rdf:resource="&a;I.4"/>
    <a:hasClassification rdf:resource="&a;I.5"/>
    <a:hasKeyword> digital image </a:hasKeyword>
    <a:hasKeyword> shape recognition </a:hasKeyword>
</a:Publication>
<a:Person rdf:ID="CSall1176"
    a:hasName="Teh-Chuan Chen"/>
<a:Person rdf:ID="CSall1177"
    a:hasName="Kuo-Liang Chung"/>
```

Figure 4. A part of the *CompuScience* knowledge base

The keywords for some publications are not provided in the original *CompuScience* dataset. In that case the keywords are generated by a NLP (nature language processing) method that extracts the frequently occurred phrases from the abstract of the publication. This method is out of the scope of this paper.

Figure 5 presents the simplified integration architecture. A user's query is executed against a full-text search engine (in this case Lucene – http://jakarta.apache.org/lucene/docs/index.html). In the case that a user requires refinement of his query, the query string is transformed into an ontology based query (the task of the "conceptualisation" module in Figure 4) and processed using the approach presented in the paper (the task of the "query refinement" module in Figure 4). The generated refinements are translated into a set of query strings and retrieved to the user.

---

[3] The ontology was derived from the given *CompuScience* database schema using the approach described in [Sto02].

[4] The public available version does not yet contain this query refinement service

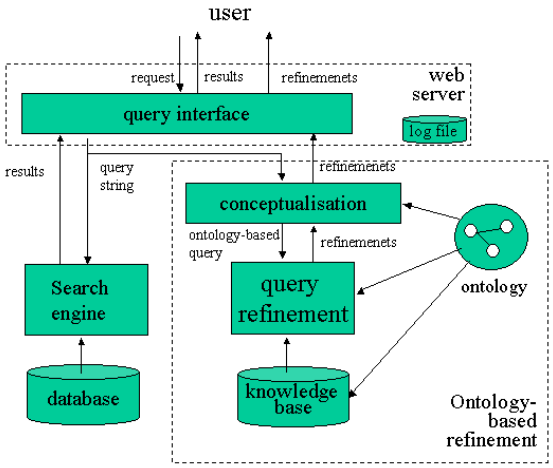

Figure 5: The integration of the logic-based refinement in a traditional information portal (a simplified model)

Conceptualisation

Lucene provides a rich query language through the Query Parser. Generally, it supports fielded data, so that a query about documents, written by a person named "john" in the year "1990", whose title contains the word "neural" and the body contains the word "networks", looks like:

+author:john +title:neural +year:1990 +body:networks.

The query can be easily mapped into an ontology query:

forall X <- X:Publication and X[hasAuthor->Y] and Y[name->Y1] and subs(Y1,"john") and X[hasTitle->Z] and subs(Z, "neural") and X[hasYear->"1990"] and X[hasKeyword->"networks"]. (1)

The above syntax is based on F-Logic [Kif95] and corresponds to the Definition 4. The predicate *subs*(*a*, *b*) returns true if string *b* is contained in the string *a* and it can be treated as a built-in feature of an ontology representation language.

The conversion from an ontology-based query to a Lucene-like query string is straightforward.

Query refinement

According to the Definition 7, the precondition for applying the logic-based query refinement operator ρ_l is the existence of an ontology-based query. It means that our approach can be directly applied on the queries in the form similar to (1). However, in this case study we can benefit from the combination of the free-text search and the logic-based search since the former is characterised by the very high recall (but low precision), whereas the second is characterised by the very

high precision. Therefore, if we apply the logic-based refinement not on the ontology-based query generated from a keyword-based query, but rather on the results retrieved by the free-text search engine, we get the opportunity to generate the candidates for the refinement (see Definition 7) from a larger set of instances. Consequently, we get more precise refinements. Moreover, since we avoid performing an ontology-based query on the whole knowledge base, we reduce the time needed for generating refinements.

From the conceptual point of view this procedure is possible, since our query refinement has a model-theoretic interpretation (see Definition 6 and Definition 7), i.e. it can be interpreted through the set of a query's results. From the technical point of view the approach is feasible, since the identifiers of the publications are shared between the database and the knowledge base.

As presented in Definition 7 the logic-based refinement generates two types of the refinement:

- based on adding new literals (e.g. the relation instance X[hasKeyword->"detection"] can be added to the query (1));
- based on the substitution of some variables (e.g. regarding query (1), the variable Y1 can be replaced with the concrete value "johnny").

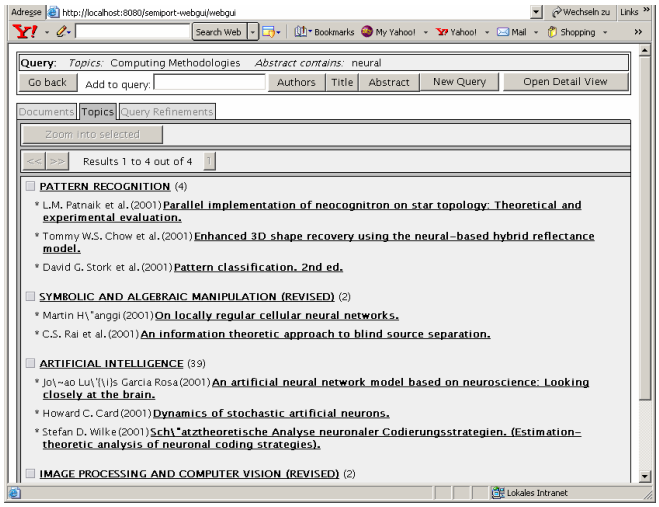

Figure 6: A screenshot from the test portal. The usage of the ACM Classification for the query refinement: the second level of the decomposition. In the first level the top-level category "Computing Methodologies" was selected

However, the hierarchical structure of some instances can be used for more structured refinements. More precisely, our approach exploits the ACM hierarchy in order to structure the refinements in a more abstract way. Consequently, a user can define a query that corresponds to his information need more easily. In that proc-

ess the ACM categories are decomposed in a step-by-step manner. The results are clustered firstly according to the top-level categories. After a user selected a category, it is decomposed on the lower levels. This process is repeated subsequently. Consequently, a user can define a query that corresponds to his information need more easily. Figure 6 illustrates this process.

The refinements are ranked according to their informativeness. From the machine learning research it is known that usefulness of an attribute (i.e. a constraint) for traversing the searching space is proportional to its information content, that is frequently measured using entropy [Wit00]. Indeed, the entropy shows the interestingness of a constraint regarding its relevance for the user's need. We extend the traditional approach for measuring entropy by introducing the concept of the variable ambiguity in order to select a variable that is the is the most informative.

Therefore, we define the suitability of each of variables in the following manner:

$$Suitability(X, Q) = VariableAmbiguity(X, Q)/Gain(X, Q),$$

where

$$VariableAmbiguity(X,Q) = \frac{|Relation(Type(X))|+1}{|Assigned\ Relations(Type(X),Q)|+1}, \text{ where}$$

Relation(*C*) is the set of all relations defined for a concept *C* in an ontology, *AssignedRelations*(*C,Q*) is the set of all relations defined in the set *Relation(C)* which appear in the query *Q*.

Gain is the standard measure of the informativeness [Wit00], i.e. for the probability distribution of the values (instances) that belong to the variable *X*, $X = (x_1, ..., x_n)$:

$$Gain(X, Q) = Info(T) - Info(X, T)$$

$$Info(X,T) = \sum_{i=1,n} \frac{|x_i|}{|T|} E(x_i)$$

where *T* is a set of all examples relevant for a query *Q* and *E* is the standard measure for the entropy:

$$E(w) = -\sum_{i \in Category} w_i * log(w_i)$$

where *Category* is the set of all categories for the classification and w_i is the distribution of instances that belong to the category *i* ($w = (w_1, ..., w_c)$).

In this way we ensure that the most ambiguous and the most informative variables are selected as the most relevant for the refinement.

Finally, Figure 7 represents the result of applying the approach presented in Section 3 on the *CompuScience* dataset. A user is provided with a complete and minimal list of refinements that can help him to refine his query according to his information need.

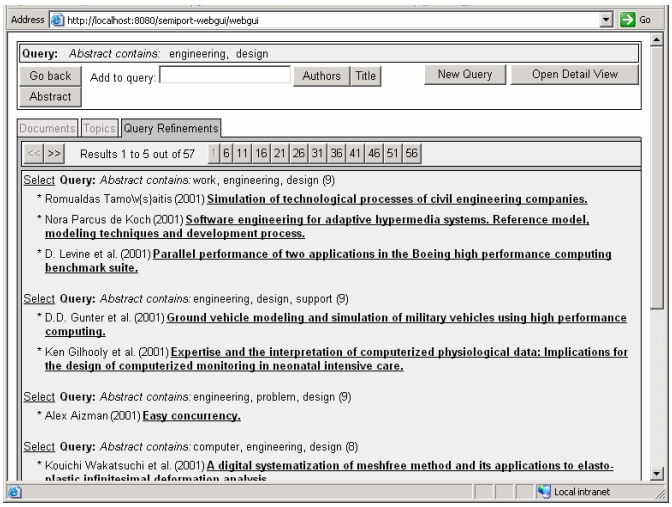

Figure 7: A screenshot from the test portal. The list of the refinements generated for the query "engineering and design"

4.2 The Evaluation

In order to prove the usability and validity of our approach we performed several evaluation studies. Due to lack of space we present here one of them: proving the usefulness of the query refinement service.

This experiment is a classical user-driven study, in which we wanted to prove the user's acceptability of the proposed approach. We compared the effectiveness of the searching with and without the query refinement approach. Since it was difficult to define in advance a set of relevant resources for a query regarding the given repository, we set-up an experiment in which each participant had to perform an unsupervised searching process. More precisely, each participant had to choose ten tasks on his own and a half of them to perform using the standard query interface and another half using the query refinement support. For each task a participant should find five relevant results[5]. In order to take into account the quality of selected relevant results, each participant had to express his *confidence* in these results. The confidence describes a participant's sureness that the selected results are the best possible ones (i.e. that there is no a better result for his need). It is measured on the scale 1 - 4, whereas 4 means maximal confidence. Beside confidence, we measured the *length of a query session* (number of querying steps) and the *duration of a query session* for each query. We selected the 15 participants, who have been undergraduate students in Computer Science. Each of them made a

5 Each participant determined the relevance in each searching session on his own

query related to the research he is familiar with. No additional instructions were given to them.

The results of the first experiment are presented in Table 2.

Method for querying	Duration of a session (in sec.)	Length of a session (num. of steps)	Confidence
Query refinement	37,28	5,58	3.23
Standard	57,30	3,87	2.26

Table 2: Results from the first evaluation study

Discussion: Although searching supported by our approach required more querying steps for a task (column 3) it is performed faster (column 2). Moreover, participants were more confident in results found by using the query refinement support in querying (column 4). This means that our approach provides refinements that are very useful (relevant) for a current query, since a user did not spend much time in a querying step. Finally, the logic-based approach covers a large part of the searching space with such refinements, so that a user is very confident with results selected as relevant, i.e. he has feeling that lots of alternatives are taken into account in the querying process. This is a very important feature in recommender applications – a user should trust the recommendation process. In order to prove if these differences can be considered statistically significant we performed a paired t-test for each measure. It did reveal the superiority of our approach with respect to all three parameters ($p < 0.0001$).

5 Related Work

Using lattices for query refinement process is not new, as some lattice representations were used in early IR [Soe67] and even more recently [Spo94] for refining queries containing Boolean operators. However as these approaches typically rely on a Boolean lattice formalization of the query, the number of proposed refinements may grow too large even for a very limited number of terms and they may easily become semantically meaningless to the user. These limitation can be overcomed by using concept lattices. In [Car98] the authors described an approach, named REFINER, to combine Boolean information retrieval and the content-based navigation with concept lattices. For a Boolean query REFINER builds and displays a portion of the concept lattice associated with the documents being searched centred around the user's query. The cluster network displayed by the system shows the result of the query along with a set of minimal query refinements/enlargements. A similar approach is proposed in [Bec01], by adding the

size of the query result as an additional factor of the navigation. Moreover, the distance between queries in the lattice is used for similarity ranking.

Conceptually, the most similar approach to our query refinement system is the Query By Navigation [Bru92], an approach for the navigation through a hyperindex of query terms. The hyperindex search engine [Bru97] aims users to add, delete or substitute a term from the initial query by providing the minimal query refinements/enlargements. It is designed specifically to (i) help user to express a precise description of their information need and (ii) reduce information overload by presenting the search result at a higher level of abstraction. Moreover, in [Gro00] the analogy between the lithoid, a crystalline structure which organizes document descriptions (and may be used to support searchers in formulating their information demands via Query by Navigation) and the formal concept lattice is shown and used in the phrase searching. However, all of presented approaches are related to Boolean queries.

In terms of the formal framework, Chaudhuri [Cha90] proposed an elegant one to describe query modification, and especially query generalization, for the relational model. He defined extended queries which express additional constraints on the answer set. Several query modification operators, mainly based on the structure of a query, are defined in order to model constraints which can be added to a query. However, the goal was not to support the refinement of a user's need, but just the extension of the query. Therefore, a generalization contains only one way to modify the query. Beside the difference in defining modification operators, we enable a step-by-step modification in which a user can define on his own which of modification can be relevant for his need. An extension of [Cha90] for the case of XML dataset can be found in [Lee02]. Recently, a framework for the refinement of SQL queries in the multimedia databases was proposed [Ort02]. Query refinement is achieved through relevance feedback where the user judges individual result tuples and the system adapts and restructures the query to better reflect the users information need. In that way a kind of similarity search is achieved. However, the approach does not treat the refinement process formally, but rather as a set of heuristics (like predicate addition or removal) described as query refinement strategies. Moreover, it does not generate a set of refinements which can support a user in developing ill-defined information needs.

Regarding searching in product catalogues the most similar approach is presented in [Ric03]. It is an extension of a mediator architecture that supports the relaxation or tightening of query constraints when no or too many results are retrieved from the catalogue. The query language is a type of Boolean queries suitable for the (web) form based querying against product catalogues. The query tightening is enabled when the cardinality of the resulted set has reached a predefined threshold and it is realized by selecting the most informative, not yet constrained product features. The information content of a feature is defined by measuring its entropy. Like previous one, this approach does not treat the problem of query refinement on an ontology-based level.

Finally, our approach can be seen as a method for Interactive Query Refinement for the case of logic-based information retrieval. In that sense our recommendations can be treated as a combination of subject thesauri and co-occurrence term lists [Sch96].

6 Conclusion

In our previous work we have defined an approach for refining (relational) queries which enables a user to navigate through the information content incrementally. In each refinement step a user is provided with a complete but minimal set of refinements, which enables him to develop/express his information need in a step-by-step fashion. The approach is based on the model-theoretic interpretation of the refinement problem, such that the query refinement process can be considered as the process of inferring all queries which are subsumed by a given query.

In this paper we presented the challenges for a information retrieval process and explained the roles an ontology can play in resolving these problems. In order to illustrate the advantages of the proposed logic-based query refinement process we presented a case study in which our approach is used as a support for the traditional (free-text based) searching process in a bibliographic information portal. It supports so the called step-by-step query refinement, which enables a novel user to inspect the content of the bibliographic database in a more systematic manner.

The evaluation study showed two main advantages of such a refinement: (i) a user can find relevant documents faster and (ii) he is more satisfied with the relevance of the documents for his information need.

Acknowledgement. Research for this paper was partially financed by BMBF in the project “SemIPort” (08C5939) and EU in project “KnowledgeWeb” (507482).

References

[Bec01] Becker, P. and Eklund, P.: Prospects for Document Retrieval using Formal Concept Analysis, Proceedings of the Sixth Australasian Document Computing Symposium, Coffs Harbour, Australia, December 7, 2001.

[Bru92] Bruza, P. and van der Weide, T.: Stratified Hypermedia Structures for Information Disclosure. The Computer Journal 35(3): 208-220, 1992

[Bru97] Bruza, P. and Dennis, S : Query Reformulation on the Internet: Empirical Data and the Hyperindex Search Engine. In: Proceedings of RIAO97, Computer-Assisted Information Searching on Internet, Montreal, June 1997

[Car98] Carpineto, C. and Romano, G.: Effective reformulation of boolean queries with concept lattices. In Flexible Query Answering Systems FQAS'98, pp. 277-291, Berlin, Springer, 1998.

[Cha90] Chaudhuri, S.: Generalization and a Framework for Query Modification, In IEEE ICDE, Los Angeles, CA, Feb. 1990.

[DeR96] De Raedt; L. and Dehaspe, L.: Clausal Discovery. Report CW 238, Department of Computing Science, K.U.Leuven, 1996

[Eft95] Efthimiadis, E.N.: User choices: A new yardstick for the evaluation of ranking algorithms for interactive query expansion, Information Processing and Management, 31(4), 605-620, 1995.

[Go97] Godfrey, P.: Minimization in cooperative response to failing database queries, International Journal of Cooperative Information Systems (IJCIS), 6(2):95-149, 1997

[Gro00] Grootjen, F.: Employing semantical issues in syntactical navigation. Proceedings of BCS-IRSG 2000 Colloquium on IR Research, 5th-7th April 2000

[Kif95] Kifer, M.; Lausen, G. and Wu, J.: Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of the ACM , Vol. 42, 741-883, 1995

[Lee02] Lee, D.: Query Relaxation for XML Model Dongwon Lee, In Ph.D Dissertation, University of California, Los Angeles, June 2002

[Ort02] Ortega-Binderberger, M.; Chakrabarti, K. and Mehrotra S.: An Approach to Integrating Query Refinement in SQL, 2002 Conference on Extending Database Systems (EDBT), pages 15-33, March, 2002

[Ric03] Ricci, F.; Venturini, A.; Cavada, D.; Mirzadeh, N.; Blaas, D. and Nones, M.: Product Recommendation with Interactive Query Management and Twofold Similairty. In proceedings of the 5th International Conference on Case-Based Reasoning (ICCBR 2003), 2003

[Sar75] Saracevic, T.: Relevance: A Review of and a framework for the thinking on the notion in information science, Journal of the American Society for Information Science, 26, (6), 321-343, 1975

[Sch96] Schatz, B.R.; Johnson, E.H.; Cochrane, P.A. and Chen, H.: Interactive Term Suggestion for Users of Digital Libraries: Using Subject Thesauri and Co-occurrence Lists for Information Retrieval. Digital Libraries 126-133, 1996

[Soe67] Soergel, D: Mathematical analysis of documentation systems. Information storage and retrieval, 3:129-173, 1967

[Spo94] Spoerri, A: Infocrystal: Integrating exact and partial matching approaches through visualization. In Proceedings of RIAO 94, pp- 687-696, New York, 1994

[Sto02] Stojanovic, Lj.; Stojanovic N. and Volz R.: Migrating data-intensive Web Sites into the Semantic Web, ACM SAC 2002, 2002

[Sto03] Stojanovic, N.: An Approach for Using Query Ambiguity for Query Refinement: The Librarian Agent Approach, 22nd International Conference on Conceptual Modeling (ER 2003), Chicago, Illinois, USA, Springer, 2003

[Sto04] Stojanovic, N.; Studer, R. and Stojanovic, Lj, A logic-based approach for query refinement, Web Intelligence WI 2004, IEEE/ACM, 2004

[Wit00] Witten, I.H; and Frank, E.: Data Mining, Morgan Kaufmann Publisher, 2000