

# Bilingual Information Retrieval with HyREX and Internet Translation Services

Norbert Gövert

University of Dortmund  
goevert@ls6.cs.uni-dortmund.de

**Abstract.** HyREX is the *Hypermedia Retrieval Engine for XML*. Its extensibility is based on the implementation of physical data independence; its query interface on the conceptual level consists of data types with respective vague search predicates. This concept enabled us to add search predicates for the data type *text* for doing bilingual text retrieval. Our implementation uses free Internet resources for translating topics in English to German and vice versa.

## 1 Introduction

Typical Information Retrieval (IR) applications offer information to the user which consists of more than just plain text documents. Digital libraries for example do not only offer full texts of scientific publications but also metadata comprising bibliographic information as well as indexing information like e.g. subject descriptors or classification codes. Often markup languages like SGML or XML are used to expose the logical structure of documents on the one hand and the attribute structure of metadata on the other hand.

This kind of fine grained markup of logical and attribute structure should be explored by IR systems in order to offer special search predicates for different types of data. For example for searching for person names like in an author attribute similarity search for proper names should be offered. These comprise not only string search but especially the possibility to search for phonetically similar names. Accordingly for dates not only predicates for testing equality should be offered but also predicates like *greater than*, *less than*, or vague predicates like *around date*.

HyREX<sup>1</sup>, the *Hypermedia Retrieval Engine for XML*, offers such kind of search predicates for different data types. Here data types with their respective (vague) search predicates build the interface to the conceptual level and thus hide their implementation details on the physical (internal) level. The concept of data independence is further explained in Section 2. Instead of treating the different data types as being independent of each other it is more appropriate to use an inheritance hierarchy. This kind of relationship on data types is used

---

<sup>1</sup> <http://ls6-www.cs.uni-dortmund.de/ir/projects/hyrex/>

to integrate bilingual IR mechanisms into HyREX (Section 3). Translation of queries is done using rather naive dictionary and machine translation methods. In Section 4 experiments with HyREX and the CLEF 2000 collections and their respective results are described. Section 5 gives a conclusion and an outlook on further work.

## 2 Data Independence in HyREX

The general idea underlying the concept of data independence is the following: By introducing several abstraction levels for data organisation, changes at a certain level do not affect the higher levels. For example, if an index on the physical level is added for speeding up certain types of queries, this should not affect the search operations on the conceptual level, except that some of them can then be processed more efficiently.

In the ANSI/R3/SPARC model [Tsichritzis & Klug 78], originating from the field of databases, three levels of data organisation are distinguished:

- The *physical (internal) level* deals with internal data and record formats and access structures.
- On the *conceptual level*, the complete conceptual schema of the database is visible. However, *physical data independence* guarantees that any changes on the internal level do not affect any application addressing the conceptual level.
- The *external level* provides specific views of the database by referring only to those relations and attributes that are needed by a specific application.

When we designed HyREX, we adopted these concepts for data independence from the database field. HyREX deals with the physical level, that is access paths for efficient query processing are provided through a proper interface to the conceptual level. This leads to the following advantages:

- Physical data independence: Search operations are independent from the availability of access paths. In many retrieval applications one can observe that this is not the case: Most systems only allow for queries which can be directly answered from an existing inverted file. In HyREX physical data independence is reached by different levels of index support. They range from *scanning* (no index available; queries are processed by directly scanning through the documents) over *support structure* to *direct index* (for example an inverted file for term searches).
- Appropriate search operations are provided. For example in most retrieval systems noun-phrase search is based on proximity operators. Here, the user has to decide for criteria which make up a phrase (e. g. distance and ordering of constituents of a phrase in the documents text). The philosophy of HyREX in this case would be to hide such implementation details from the user. The user is provided with a specific search predicate for phrases, while the system internally decides how a phrase is defined. Of course HyREX's

decision might be based on criteria like distance and ordering but also more enhanced methods can be implemented without affecting the user's search interface.

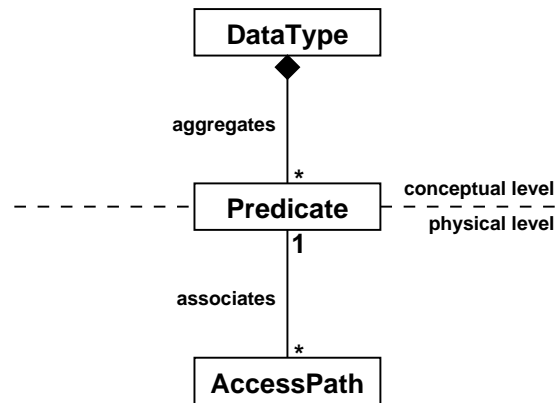
- Finally, the concept of data abstraction by means of different system levels helps to modularise the system. HyREX has an object-oriented design.

In HyREX the search interface is made up of data types with vague predicates. The attribute structure of a given document base is therefore mapped onto a schema which assigns each attribute its respective data type:

$$Schema := \{AName_1 : Datatype_1; \dots; AName_n : Datatype_n\}$$

For example a simple schema for a literature database could look like the following:

$$Schema := \{Author : PersonName; Content : Text; PubDate : Date\}$$



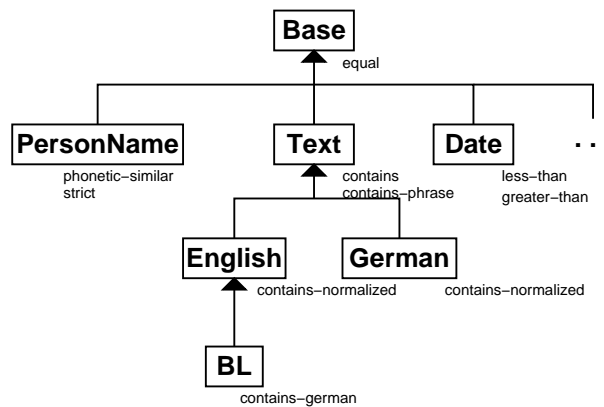
**Fig. 1.** General UML class diagram [Fowler & Scott 97] of a data type: a data type aggregates one or more search predicates. These predicates are implemented by and therefore relate to one or more access path structures.

A data type is made up by its domain (i. e. values comprising the data type) and appropriate (vague) search predicates, which can be applied to elements from the data type's domain (a more formal view on data types and search predicates is given in [Fuhr 99]). Figure 1 shows the general UML class diagram of a data type. The data type aggregates one or more search predicates. At the search predicates we separate the conceptual from the internal level: While the predicates make up the search interface from the conceptual level their implementation by means of appropriate access paths or scanning is hidden on the physical level. Details of the implementation are given in [Fuhr et al. 98].

Having the schema which assigns each attribute a data type with their respective predicates one can formulate queries at the conceptual level. Such queries basically are triples consisting of an attribute name, a predicate, and a comparison value. W. r. t. the schema above, for example the following queries can be issued:

- *Author* **sounds-like** *Norbert Fuhr* asks for documents being authored by someone who’s name sounds similar to *Norbert Fuhr*.
- *PubDate* **around-year** *1999* asks for documents which have been published around year 1999.
- *Content* **contains-phrase** *probabilistic IR* asks for documents dealing with the concept *probabilistic IR*.

Instead of treating the different data types as being independent from each other it is more appropriate to use an inheritance hierarchy. That is data types can inherit from each other. A data type  $D'$  which is a specialisation of a data type  $D$  inherits all predicates of  $D$  and can be extended by more specific search predicates. A simple inheritance hierarchy is depicted in figure 2: while for example the data type *Text::English* inherits from its ancestors the predicates **equal**, **contains**, and **contains-phrase** it specialises the *Text* data type by the language dependent **contains-normalized** predicated, which provides for searching for word stems.



**Fig. 2.** Inheritance hierarchy on data types.

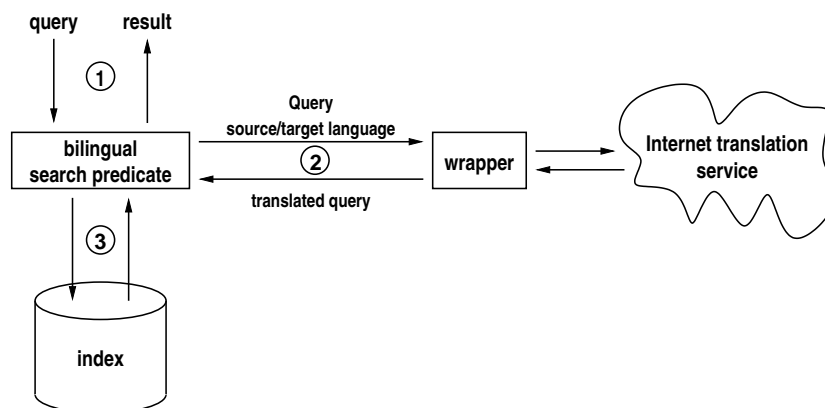
### 3 Search Predicates for Bilingual Retrieval

Having a system which is extensible w. r. t. data types and their respective search predicates we decided to extend the *Text::English* and *Text::German* data

types by search predicates for bilingual text retrieval. These predicates needed to perform the translation of topics and queries from German to English in case of data type `Text::English` and vice versa in case of data type `Text::German`.

For translation of queries we adopted two rather naive, but fully automatic approaches. In both approaches we used free Internet resources:

- Approach 1 uses the Babelfish translation service<sup>2</sup> of Altavista. This service allows to translate passages in a source language to a given target language. Besides the translation from German to English and vice versa, Babelfish is capable of various other languages.
- Approach 2 uses an ordinary online dictionary for word-by-word translations. We chose the Leo Dictionary service<sup>3</sup> for this purpose. Leo provides for a English / German dictionary with about 223 900 entries. Translations can be done in both directions. Since also composed words and phrases are included in the dictionary, we exploited this by not translating the original topics word-by-word but by interpreting each two neighbouring terms as phrases. Adopting a real naive approach we even didn't take measures in order to tackle the word disambiguation problem.



**Fig. 3.** Bilingual search predicates, implemented using free Internet translation services.

Figure 3 shows the general scheme of our search predicates for bilingual text retrieval. The user gives the query in a source language, which is translated by means of a translation wrapper. The task of the wrapper is to give a uniform interface to free translation resources on the Internet: It accepts the query as given by the user plus source and target language and then handles the translation through the service it was implemented for.

<sup>2</sup> <http://babelfish.altavista.com/>

<sup>3</sup> <http://dict.leo.org/>

## 4 Experiments

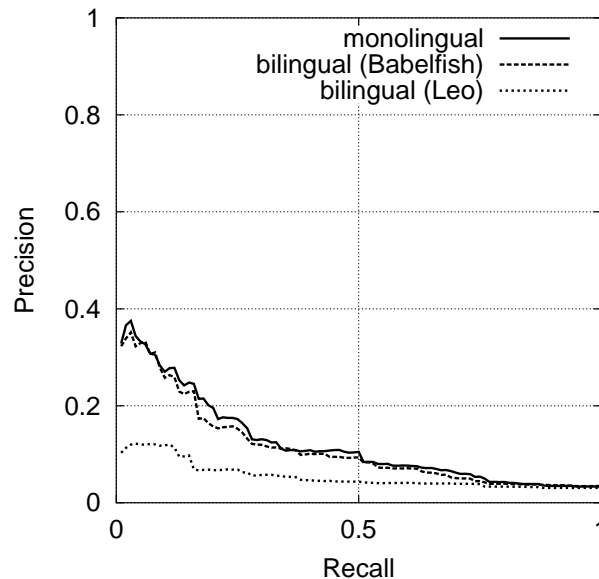
For evaluating our search predicates for bilingual retrieval in terms of effectiveness we used two document collections from the *Cross-Language Evaluation Forum*<sup>4</sup>. Both, the *la\_times* collection and the domain-specific *GIRT* collection come with topics in German and English; relevance judgements have been derived by judging the results of the CLEF 2000 participants.

Both test collections have been indexed by HyREX; to build the proper access paths for the bilingual search predicates we applied language specific stop-word removal and stemming on the documents' content. The well-known  $tf \times idf$  scheme [Salton & Buckley 88] has been applied for term weighting.

For comparison we also performed monolingual retrieval runs on both collections. Effectiveness has been measured in terms of recall and precision, the results are presented by means of recall-precision curves and the average precision w. r. t. 100 recall points.

### 4.1 *la\_times*

The *la\_times* collection consists of 84 347 newspaper articles in English from the 1994 Los Angeles Times<sup>5</sup> volume.



**Fig. 4.** Effectiveness of bilingual retrieval with the *la\_times* collection

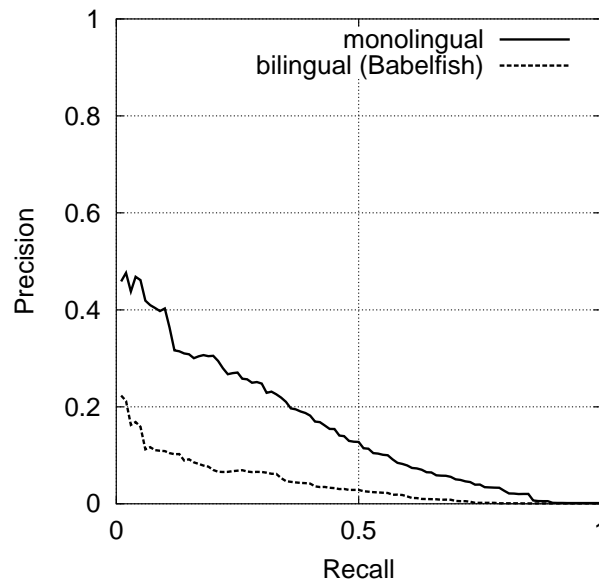
<sup>4</sup> <http://www.iei.pi.cnr.it/DELOS/CLEF/>

<sup>5</sup> <http://www.latimes.com/>

For bilingual retrieval on the *la\_times* collection both, the Babelfish and the Leo approach have been used to translate the topics. Figure 4 shows the recall-precision curves resulting from the bilingual (german to english) and monolingual retrieval runs. The average precision is 11.23 % for the bilingual run using the Babelfish approach, 5.44 % for the bilingual run using the Leo approach, and 12.11 % for the monolingual run.

## 4.2 GIRT

The *GIRT* (German Indexing and Retrieval Test database) collection provides for 76 128 documents from the *social sciences* domain. The documents are in German and have been put together by IZ Bonn<sup>6</sup>. Topics were given both in English and German.



**Fig. 5.** Effectiveness of bilingual retrieval with the *GIRT* collection

For bilingual retrieval on the *GIRT* collection the English topics have been translated by the Babelfish approach. Figure 5 shows the recall-precision curves resulting from the bilingual (German to English) and monolingual retrieval runs. The average precision is 4.20 % for the bilingual run and 15.78 % for the monolingual run.

<sup>6</sup> <http://www.bonn.iz-soz.de/>

### 4.3 Analysis

The results show that bilingual retrieval implemented through free Internet translation services can be employed to domain-unspecific information retrieval applications. In case of the Babelfish approach together with German-to-English bilingual retrieval we yielded effectiveness which is comparable to the effectiveness reached by monolingual retrieval. However, the same approach did not perform comparably well on the domain-specific GIRT collection.

Considering the Leo approach which used a word-by-word translation of the original topics one can say that this approach is too simplistic. Without any means of word disambiguation a reasonable effectiveness could not be reached. During the translation process the size of the topics has grown by 92 % on average (on average the original topics contain 20.12 terms, while the topics translated by Leo consisted of 38.73 terms).

## 5 Conclusion

We have used HyREX for performing bilingual information retrieval. While the overall performance of the system in terms of effectiveness is rather low, we have shown that the system's design and its flexibility allows us to extend it by cross-lingual IR methods. The architecture of HyREX, which provides data types with vague predicates as an query interface on the conceptual level, forms the basis for these extensions.

Our next steps will aim at improving the retrieval effectiveness. More enhanced methods for bilingual IR need to be implemented, especially for retrieval in domain-specific collections. Furthermore we would like to further extend the system in order to be able to also participate in multi-lingual retrieval tasks.

## References

- Fowler, M.; Scott, K.** (1997). *UML Distilled. Applying the Standard Object Modeling Language*. Addison Wesley, Reading, Mass.
- Fuhr, N.** (1999). Towards Data Abstraction in Networked Information Retrieval Systems. *Information Processing and Management* 35(2), pages 101–119.
- Fuhr, N.; Gövert, N.; Rölleke, T.** (1998). DOLORES: A System for Logic-Based Retrieval of Multimedia Objects. In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 257–265. ACM, New York.
- Salton, G.; Buckley, C.** (1988). Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management* 24(5), pages 513–523.
- Tsichritzis, D.; Klug, A.** (1978). The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems. *Information Systems* 3(3), pages 173–191.