Semantic Search in LeActiveMath

Paul Libbrecht, Erica Melis
DFKI, Saarbrücken, Germany

Abstract. The web, as we experience it nowadays, is heavily based on search engines such as Google or Yahoo!. These engines are the essential step to discover web-content that would, otherwise, only be available after too many clicks. The field of computer-science which serves as their theoretical basis is information retrieval. However, the main focus of information retrieval is on textual content, that is words and sentences. Little research has been done, however, both in terms of research or tools, for information retrieval regarding *mathematical* content on the web as can be seen, for example, in the overview [M05].
In this article, we present work done for the LeActiveMath learning environment which stores and presents semantically encoded mathematical content. We have adapted information retrieval techniques to this semantic content in order to offer to learners reasonably tolerant searchability for text, metadata, and formulæ. Our efforts follow information retrieval principles stating the essential needs for fast response and easy query inputs.

1 Introduction

ActiveMath is an intelligent web-based learning environment for mathematics, it presents semantically encoded mathematical documents to learners, and allows them to practice by doing interactive exercises.[1] ActiveMath uses a fine-grained knowledge representation of mathematical documents based on the `OMDoc` encoding [OMDoc1]. Using it, the learners' competencies can be modelled and adaptive behaviours, such as the choice of content needed to achieve a learning goal, can be provided. The EU project LeActiveMath, Language Enhanced ActiveMath, developing a larger framework and content collection around ActiveMath. Within this project, the plain-text search engine available earlier in ActiveMath has been refined to bring semantic search capabilities to users which is the focus of our article.

Requirements analysis done among the partners of the project, including representative stakeholders of the publishers' and teachers' communities, has indicated that:

[1] More information about the ActiveMath learning environment can be seen from the project's home page: `http://www.activemath.org/`.

– the search tool of LeActiveMath should be very easy to use but should allow querying for text, meta-information, as well as mathematical formulæ.
– the search queries should be *reasonably tolerant* as one expects frustration to arise in a tool doing simple exact matches
– the search tool of LeActiveMath should offer access to all mathematical content thus supporting explorative learning

We expect the search tool to be used by several types of users of LeActiveMath who differ in their expectations and proficiency:

– the beginner learner uses it only for the purpose of searching quickly for a reminder or to discover new content. He will search mainly for words and for formulæ only if he can copy and paste them. The search interface and presentation should be simple and straightforward for him.
– the advanced learner may want to see details about each item and maybe search variants of presented item and search for mathematical formulæ or items' characteristics.
– an author may want to see all items, including the description of OpenMath symbols, and see details about each.

1.1 Usage of LeActiveMath Knowledge Representation

LeActiveMath stores and presents content encoded in the `OMDoc` xml-language added with metadata for educational purpose [LeAM-D6]. It is built on content split in units which are called *items*: definitions, theorems, exercises... Items are addressable by unique identifiers. Items have a granularity well-suited for our search purposes: they are easily identifiable and recognizable by a user as a separate entity and can be, partially, taken out of context.

Therefore the search engine searches for items and present results one item at a time, including relations from this item to other items.

1.2 Information Retrieval for LeActiveMath

Following classical information retrieval vocabulary [vR79], `OMDoc` items are defined as the *documents* of our search function. Based on the *resonable tolerance* requirement, we expect the search results to be often too numerous to be easily overviewed. Information retrieval indicates that a good paradigm is to provide search *ranking* where a score is computed for each matched item indicating how *relevant* the match is and to present search results from highest score on; additionally users are more likely to experiment with search and get results quickly than take the time to wait for quality search-results [vR79].

Information retrieval has been mainly focussing on the retrieval of text documents and on textual search. Two essential ingredients are put to use:

– An *analysis* or *tokenization* process which converts the text documents that will be searched in a sequence of *tokens*. Tokens are, typically, words, but these may be the result of a translation which, for example, removes the "s" character of plural words in English.
– An index which stores the occurrence of tokens in documents and can be efficiently searched for. Queries are formulated as the search for documents where tokens (results of the same analysis process) occur.

2 Prototype Description

The LeActiveMath search is a prototype that is part of the LeActiveMath learning environment. In this section, we describe the essential ingredients of the search engine.

2.1 Core Index

We have chosen the Lucene library[2] to maintain the core index. This library is a recognized open-source library and is in use in many industrial strength systems (such as Wikipedia or Technocrati). It provides storage and indexing of documents and high-speed queries on this index delivering the documents along with query-match ranking. The Lucene library is also the base of the content storage engine of the current ActiveMath *LuceneMBase*.

In our search, the documents are the items of the content. To build the index, we process the following information for each item:

– the titles
– the metadata information
– the textual content
– the mathematical formulæ in the textual content

This section describes how the index is built based on this information, that is, mostly, how texts in various languages and mathematical formulæ are tokenized. Along this description, the possible low-level queries are presented. It presents an overview of how each query is *boosted*, that is, is given a weight, in order to enter the computation of the rank of a found document, and concludes with an example illustrating all aspects.

[2] See `http://lucene.apache.org/`.

Storage of metadata information Each `OMDoc` item can have a metadata element. The metadata element is the place to store *attributes* of each item. For example, the field of study for which this item is intended (e.g. physics, law, ...) or the time one expects a learner will need to read the item. Metadata attributes as a list of name-value pairs. Documents can be queried for using the same name-value pairs.

The metadata of `OMDoc` items is also the container of relations between items, for example, the fact that a definition depends on a proof. Since the content storage of ActiveMath already has query facilities for these relations, the index does not consider them.

Tokenization of text Following classical retrieval [vR79], the tokens are made of words of the text; these words are, before, converted to lower-case, *stemmed*[3], and too common words are removed. This tokenization is language specific, we use the classical stemmers of Porter [Por05] which was written for English and generalized to many other languages. To respect the various languages, we encode the token-stream of each language in a different field of the index. To support different ranking for matches in the title compared to matches in the text, these two fields are indexed separately.

Towards Fuzzy Matching of Text In order to support learners in their searches one has to cope with words that are misspelled and to provide results that were approximately matched (fuzzy matching).

Fuzzy matching can be done with the index data described above: the Lucene library offers fuzzy matching based on the *edit distance*, that is, it allows elementary modifications of the token's characters (add or remove a letter, permute two, ...) which is then matched with lower score. The results of this form of fuzzy matching yields sensible results most of the time, but sometimes leads to surprises such matching "class" when searching for "flash", It is well suited to match words that have been misspelled, either at query or authoring time.

In order to provide an alternate form fuzzy matching of text, a phonetic tokenization of the text is stored using the metaphone phonetic algorithm. This phonetic-tokenization algorithm is an enhancement of the original Soundex algorithm of Russell and Odell typically used in spell-checkers, it translate words having the same *sound* to the same tokens. The libraries we have currently found work for English and German, and will be tested for their applicability in Spanish.

[3] The action of stemming a word is to take it to its root so that declinations of the same words end up being the same token. For example, *groups* is stemmed to *group*

Tokenization of mathematical formulæ As any information retrieval library, Lucene understands linear sequences of tokens. One wishes, however, to query the mathematical formulæ with their structure. ActiveMath uses the OpenMath standard [OM2] which organizes mathematical objects as trees of symbols and applications. For the mathematical formulæ in texts, the sibling order of xml-tree-walks produces a sequence of tokens that no sentence could produce. The anlysis process tokenizes the OpenMath-application with a depth indication, as well as the symbols, strings, floats, and integers of OpenMath: For example, the formula $\sin x^2$ becomes:

```
_(_1                <OMA>
_OMS_transc1/sin     <OMS cd="transc1" name="sin"/>
_(_2                   <OMA>
_OMS_arith1/power        <OMS cd="arith1" name="power"/>
_OMI_2                   <OMI>2</OMI>
_OMV_x                   <OMV name="x"/>
_)_2                   </OMA>
_)_1                 </OMA>
```

Using this tokenization, we can query exact formulæ by an exact phrase match, that is, a match for a sequence of tokens. As a given expression can occur at any depth of a mathematical expression, exact phrase queries have to be expanded as a disjunction of queries for each depth.

formulæ with blanks can also be queried for: the example above would be matched by a query for the following sequence of tokens expressing the search for the sine function applied to any argument would query for the tokens

```
_(_1 _OMS_transc1/sin * _)_1
```

where the `*` indicates a blank in the phrase query which matches anything as far as the remaining part is matched.

2.2 From User Queries to Ranked Matches

We have described how the index is built from tokens and how it can be queried and matched. Let us summarize how weight is assigned to fields so that those matches that we expect are the *most important ones* are given the highest rank. Queries will be made for text, for mathematical formulæ, and for metadata attributes. These user-level queries are translated to disjunctions of index-level queries each being given a boost-factor which influences the overall score of matches. The latter is used to order the results.

This heuristics is prepared for LeActiveMath and may be tuned depending on the results of the evaluation. The list below describes boost-factors of each query-types which get multiplied if a match of the same query occurs.

– textual and mathematical matches are expanded into queries in title and in text: a match in the title of an item count twice as much as matches in the item's text.
– exact text-matches and exact formulæ matches have factor 2.0
– metadata and keyword matches are boosted by 50
– fuzzy phonetic matches are slightly less boosted by a factor of 0.8
– formulæ matches with blanks have a boost decreasing with the *length* of the matched blanks
– fuzzy matches with edit distance are boosted depending on the amount of changes (so that a single change, which is probably a typo, yields a match with a score close to an exact match whereas a radical change yields a score close to zero)

2.3 Example Tokenization of an Item And Related Queries

We present a small example of an exercise with English text and title along with a formula:

Trigonometric exercise *Let us assume* $x < y$.

Indexing decomposes the content of this item in the fields `title-en`, `text-en`, and `text-phonetic-en`:

```
attr:              type:exercise
title-en:          trigonometr exercis
text-en:           let us assum _(_1 _OMS_relation1/lt _OMV_x _OMV_y _)_1
text-phonetic-en : LT US B ASMN
```

With this content in the index, the following queries can be performed:

– textual query: if the user enters "trigonometry", tokenization of the user-query converts this word, among others, to a query for token "trigonometr", which is exactly matched to the title yielding score 10.0.
– textual-fuzzy: user enters "asuming", tokenization converts it, among others, to a query for the token "ASMN" in the *text-phonetic-en* field which is matched to the content of the phonetic english field (with score 0.8)
– metadata query: a query of the user for the type exercise would be reformulated as an index-query for the token `type:example` in the field `attr` which is matched to our item with score 1.0.
– mathematical query: if the user queries for the formula $x < y$, the query is translated into a query for
`_(_1 _OMS_relation1/lt _OMV_x _OMV_y _)_1`
in the field `text-en` which is exactly matched to our formula yielding score 1.0.

2.4 The Search-Tool

The search tool uses the pre-processed content and search techniques described in the previous section. We now present how the search facility is offered to learners using ActiveMath and the ease of use of the search user-interface.

Search input The search-tool can be activated by the input of search-words in the text-field placed for this in the menu. This produces the results in the search-window. Such a search produces the default queries: fuzzy matching (both phonetic and edit-distance fuzzyness) of text in the learner's language for concepts in the current book.

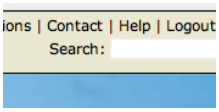

Fig. 1. search-field in the menu

The search-window enters the *plain* search mode, a mode where the search is displayed as a single string as in classical web search engines.

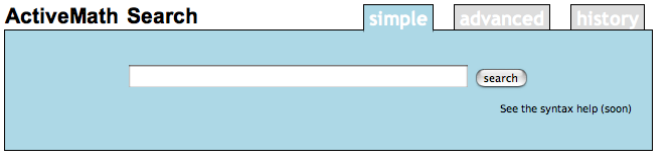

Fig. 2. The plain search, seen when first opening the LeActiveMath search

In addition, a more elaborate syntax can be used to require or exclude some word, change language, or query characteristics of the items.[4]

Searches can also be input using the *advanced search* (see figure 3) form which allows a combination of queries for:

- text queries with or without fuzzyness and exact phrases, input within a text-field
- mathematical expressions input using the Wiris input editor[5] which allows graphical input of formulæ as well as allows copy and paste of mathematical formulæ from the content.
- item characteristics as can be found in the metadata of each items, entered using pop-up menus.

[4] More details on the query-syntax, mostly intended to authors or advanced users, can be read online in the plain-text search-mode.

[5] More about the Wiris input-editor can be read from `http://www.wiris.com/`.

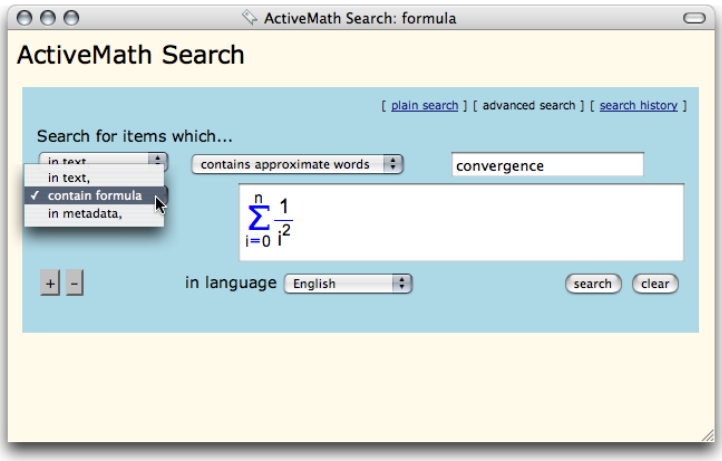

Fig. 3. The advanced search form illustrating a combinaed mathematical formula query and metadata query.

Search result presentation The results, sorted by the scores provided by the matches, are presented by title which, once clicked, show the full presentation of the item. Mastery-bullets indicate the estimated learner's *mastery* for this item. Links allow the learners to go to other search pages.

Only a first page of search results is presented if more than 20 items are found. This measure is approximately the amount of items that can fit vertically on a screen and allows a fast presentation of search results to allow further explorations of the user.

Integration of relevant web-sources In order to enable search of other mathematical web resources and to compare results of the searches, the result presentation adds links to submit the same query to search engines and content collections such as the Google engine searching the Web[6], the Wikipedia collaborative encyclopedia[7], or and the MathWorld encyclopedia[8].

Currently, these links can only be presented if the query is textual since the sources do not support metadata or semantic mathematical markup.

This integration supports exploration and, at the same time, makes the user aware of the sources which will support the user's critical thinking in terms of trust and proficiency.

[6] The Google engine is at `http://google.com`

[7] The Wikipedia encyclopedia is at `http://www.wikipedia.org`.

[8] The MathWorld repository is an enterprise of Wolfram Research Inc. and can be reached at `http://www.mathworld.com/`.

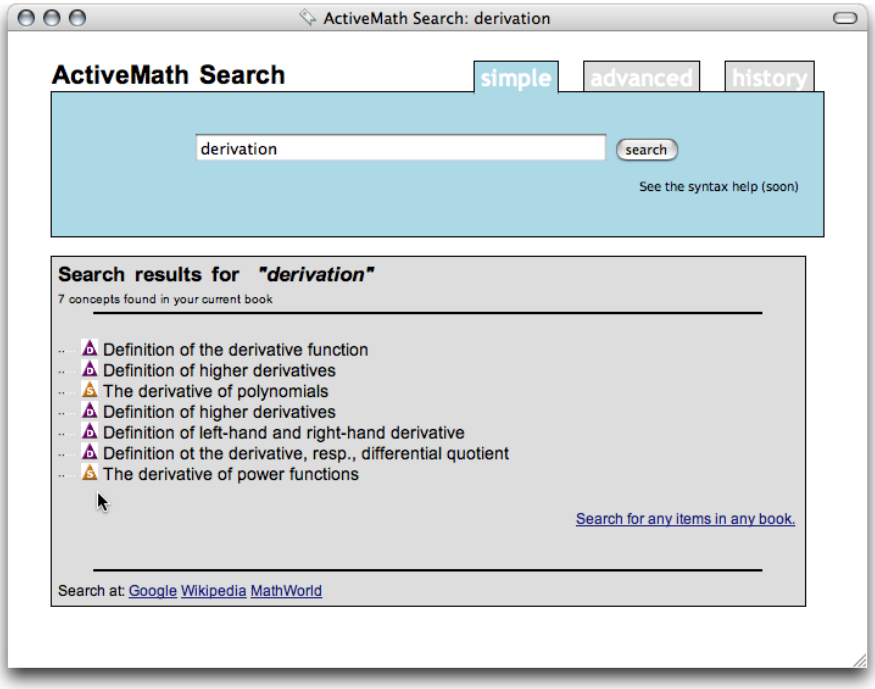

Fig. 4. Searching for derivation and selecting a concept.

Search history and search states Each search and all results obtained and browsed are stored in a search history. The LeActiveMath search engine tries to store all states of the search tool user interface. This includes the input-element under current focus in the search form, the search mode, the result page, the item being currently viewed and details about its view. Each of these states are numbered sequentially and can be re-invoked later, for example, once the search-window is closed then re-opened. The history view presents a user readable text of each query along with the titles of items shown. Each step can be restored and continued with. This contributes to make the LeActiveMath search similar in availability and memory to an opened dictionary on your desk: one can forget about it but taking it back will restore its last state.

Item display Once a link in the search result list is clicked the search tool displays single items aside of the search results. Single items display present the type, title, content of the item, the notes and *mastery* icons as well as the copyright link.

Items display is complemented with information from the metadata: in its simple form, the *for* relations from and to this items are displayed, for example *exercise*

for this item or *concept that this examples illustrates*. Pressing a *more* link opens a detailed view of the item's metadata including all relations from and to this items and all references to this item. The presentation of these relations allow the learner an explorative navigation of the rich structure of the content.

Item display LeActiveMath is using the presentation architecture of LeActiveMath [ULWM04]. That is, it uses the same rendering engine and the same notations, providing a consistent appearance of the mathematical knowledge.

3 Related Work

We briefly list the related work about the retrieval of mathematical documents in order to situate our research:

- the MBase [FK00] project has inspired both the search function and our content-storage. It offers a prototypical pattern matching of mathematical formulæ with variables being instantiated by any sub-expression: for example, querying $f(A, B) = f(B, A)$ as search for any commutativity statement. This project was based, mostly, on serial search along all OpenMath objects stored in the database which is not a very scalable approach. We have indicated in section 2.1 how our mathematical tokenization can match mathematical expressions with variables being replaced by subtrees; we have not achieved, however, the ability to compare two instances of instantiated variables (for example when querying for $f(A, B) = f(B, A)$ to match any commutativity statement requesting that the terms A and B be matched consistently). This form of query can be rephrased a query with joins and deep-equality. Experts in the field of XML databases, when asked about this form of matching seem to indicate no other strategy than serial searches.
- the MathQL project[9] who's goal is to search the CoQ-library has put forward great goals in their mathematical search such as the application of *unification* as opposed to *pattern-matching*. Such goals seem not reached yet.
- MoMM [Urb04] is a recent attempt at using formal rewriting rules to search the Mizar Mathematical Library. We have not been able to identify the feasbility of bringing such interreduction rules into the content of LeActiveMath.
- private discussions with the developers of the Digital Library of Mathematical Functions at the NIST institute indicate that search is being developed in their project. The search implementation seems to be based on a translation of the TEX sources of the library and is helped by a large number of heuristics providing a form of fuzzy matching.[10]

[9] MathQL is a subproject of the HELM project, for more information: `http://helm.cs.unibo.it/mathql/`.

[10] See `http://dlmf.nist.gov/` about the Digital Library of Mathematical Functions.

– The company Design Science Inc. has started an NSF-funded project on mathematical search and are currently running evalutations of tools where they can search MathML-presentation-encoded formulæ.[11]
– Paul Cairns has experimented with Latent Semantic Indexing in [Cai04] on the Mizar mathematical library. His study seems to carry interesting results with a very reasonable computational power.
– the thesaurus.maths.org project at Cambridge University has similar target audience than our project. It is to be noted, however, that the thesaurus only offer textual search having a TeX-based content-representation and semantic annotations only for the navigation between items.

4 Future Work

The LeActiveMath search tool provides a sturdy basis for information retrieval of `OMDoc`-encoded content. The search tool of LeActiveMath will enter in the evaluation phase of the project and will be polished and refined accordingly. Among others, the evaluation will measure the *understandability* of the search engine in comparison to other search engines.

The usage of latent-semantic-analysis as described in [Cai04] is probably worth following as much of the infrastructure that we have developed enables the models of vector-based occurrence-representation which is at the basis of this domain. The engine should measure items *close to* the queried document by using a distance based on co-occurrences of tokens, both textual and mathematical. Care has to be taken, however, as the latent-semantic-indexing process is patented and only available in a relatively impractical library.

The presentation of search results will be enhanced most probably. The simple display of mastery-bullets, item-types, and item titles may prove to be insufficient (for example, many exercises simply bare the title *Exercise*). We should consider avenues, such as the *result-gisting* approach presented in [CKS05] which display which other queries a given result-document could also match.

Graph-based navigation of the items' relations has been requested several times and seems to provide an intuitive representation of the navigation through the knowledge. We expect to embed such a navigation for learners.

[11] More about the search project of Design Science can be found `http://www.dessci.com/en/reference/searching/`.

5 Open Issues

We have not been able to assess how much interreduction of formulæ, as in [Urb04], or search-query unification as promoted by the MathQL project, could help a learner in his search. On the one hand, many rewrite-rules appear to be obviously needed (for example, applications of the associativity or commutativity). On the other hand most of these rewrites actually inject knowledge into the search which may surprise the learner. As an example, searching for the statement of the derivation of a given function could be, naturally, rephrased as the search for any function whose indefinite integral is the indicated function which may leave the learner quite perplex.

Finally we wish to stress the integration of the external search engines within the display of search results. They provide alternatives to users and also provide comparisons to them: one will be able to evaluate the quality of the search engines but also the quality of the searched content. Such resources as Wikipedia or Thesaurus.maths.org are more appropriate for the search-and-browse paradigm than the content of the LeActiveMath books since the latter were, mostly, written for a book usage. For example, in many tests we have made with single words, the corresponding search in the Wikipedia encyclopedia reached a *disambiguation* page which is a manually authored page branching to the many possible interpretations of a single word. There is no such item types within LeActiveMath knowledge representation yet. Experiments and comparisons will enable us to infer the essential differences.

References

Cai04. Paul Cairns. Informalising formal mathematics. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *Proceedings of 3rd Int. Conf on Mathematical Knowledge Management*, volume 3119 of *LNCS*, pages 58–72. Springer-Verlag, 2004. Available at `http://www.uclic.ucl.ac.uk/paul/research/MizarLSI.pdf`.

CKS05. Karen Church, Mart Kean, and Barry Smyth. Towards more intelligent mobile search. In *Proceedings of the conference IJCAI-05*, 2005. Available from `http://www.ijcai.org/papers/post-0135.pdf`.

FK00. A. Franke and M. Kohlhase. MBase: Representing mathematical knowledge in a relational data base. In F. Pfenning, editor, *Proc. 17th International Conference on Automated Deduction (CADE)*, Lecture Notes on Artificial Intelligence. Springer-Verlag, 2000.

OMDoc1.] Kohlhase, M. 2000 `OMDoc`: Towards an OPENMATH representation of mathematical documents (Seki Report SR-00-02). Fachbereich Informatik, Universität des Saarlandes. (`http://www.mathweb.org/omdoc`)

LeAM-D6. LeActiveMath Partners. D6: Leactivemath structure and metadata model. LeActiveMath Deliverable D6, The LeActiveMath Consortium, December 2004. See `http://www.leactivemath.org/index.php?id=4915`.

M05. Robert Miner. Enhancing the Searching of Mathematics, A position paper based on the proceedings of the Enhancing the Searching of Mathematics Workshop. June 8th, 2004. See http://www.dessci.com/en/reference/searching/math-searching.htm.

OM2. Stephen Buswell, O. Caprotti, D. Carlisle, M. Dewar, M. Gaëtano, and Michael Kohlhase. The OpenMath Standard, version 2.0. The OpenMath Society, 2004. Available at http://www.openmath.org/.

Por05. Martin Porter. The Porter stemming algorithm, 2005. See http://www.tartarus.org/~martin/PorterStemmer/.

ULWM04. C. Ullrich, P. Libbrecht, S. Winterstein, and M. Mühlenbrock. A flexible and efficient presentation-architecture for adaptive hypermedia: Description and technical evaluation. In Kinshuk, C. Looi, E. Sutinen, D. Sampson, I. Aedo, L. Uden, and E. Kähkönen, editors, *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT 2004)*, pages 21–25, 2004.

Urb04. Josef Urban. MoMM - fast interreduction and retrieval in large libraries of formalized mathematics. In *Proceedings of the ESFOR 2004 workshop at IJCAR'04 available at http://www.mpi-sb.mpg.de/~baumgart/ijcar-workshops/proceedings/PDFs/WS5-final.pdf*., 2004. More information on the project at http://wiki.mizar.org/cgi-bin/twiki/view/Mizar/MoMM.

vR79. C.J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979. Available at http://www.dcs.gla.ac.uk/~iain/keith/.