MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Evaluation of Mathematics Retrieval

DIPLOMA THESIS

Martin Líška

Brno, 2013

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Martin Líška

Advisor: assoc. prof. RNDr. Petr Sojka, Ph.D.

Acknowledgement

I would like to thank my supervisor assoc. prof. RNDr. Petr Sojka, Ph.D. for his guidance and advices on the topic throughout this work. I would also like to thank the members of MIR team at Faculty of Informatics, Masaryk University for their supportive work that helped the whole project to progress faster. Also thanks goes to James Edward Thomas, M.A. for his linguistic guidance as well as to everyone who helped me with the work in any way.

Work that is described in this thesis was partially co-funded by the European Union through its Competitiveness and Innovation Programme (Information and Communications Technologies Policy Support Programme, "Open access to scientific information", Grant Agreement no. 250,503).

Abstract

The thesis deals with the evaluation of mathematics information retrieval (IR). It gives an overview of the history of regular IR evaluation, initiatives that are engaged in this field of research as well as most common methods and measures used for evaluation. The findings are applied to the specifics of mathematics retrieval. This thesis also summarizes the state-of-the-art of MIaS math search system, which is already being used in an international web portal. Latest developments aiming towards the second version of the system are described. In addition to participating in the international evaluation conference and workshop, MIaS is tested for effectiveness and efficiency in this work. Measured performance indicators are evaluated and future work is suggested accordingly.

Keywords

information retrieval, IR, mathematics, search, MIR, MIaS, precision, recall, formula, collection, IR history, relevance judgment, query, NTCIR, EuDML

Contents

1 Introduction

In the world of the Internet and World Wide Web, which offers a tremendous amount of information, an increasing emphasis is being given to searching services and functionality. The founder company of the best known and most widely used search portal, Google, was rated as world's fourth biggest software company[1].

Currently a majority of web portals offer their own searching utilities, be it better or worse. These are able to search for the content within the sites, mainly text – the textual content of documents, annotations as well as the metadata of various types of information. Searching for any other differently structured data will not be very successful or is not currently supported.

Mathematical documents contain precisely such information. They are mainly collected and offered to the general public through digital mathematics libraries (DMLs). Such documents are of a scientific character and the mathematics they contain hold significant information for the meaning of the whole text. In most cases mathematics formulae cannot be described and searched for in a few words, rather these formula supplement the meaning of the texts. This indispensable information can not currently be searched for.

It is necessary to provide math-aware searching facilities within portals that provide users with substantial amounts of mathematics. Its contribution to the users of these portals and to the general public is undeniable, both in the short and long-terms [27]. There are, however, several inherent obstacles. Firstly, there is a wide diversity of mathematical notation formats used to encode math information in the content. Scientists are used to straightforward and powerful notation of TEX and its extensions, despite there being several structured formats standardized for math information interchange between machines, such as MathML. Secondly there can be significant ambiguity in written mathematics, where one mathematical expression can have several meanings. Such formulae need to be disambiguated, possibly through semantic information, a practice that almost no author observes this when writing a paper. On the contrary, a formula can be written in a countless variety of ways while still conveying the same mathematical meaning.

There were several attempts to address these issues in the past which led to the design and implementation of a solution for mathematics retrieval: MathDex, LeActiveMath, MathWebSearch, LATEXSearch, EgoMath, MIaS. Each of these employ different approaches. The most distinct of these is MathWebSearch as it is not based on full-text indexing. The systems differ in the type of processed documents, internal representation of the data, query types and

1. Forbes Global 2000, 2010. 4 Jan. 2013 `http://www.forbes.com/lists/2010/18/global-2000-10_The-Global-2000_Rank.html`

their languages [9]. There is no commonly accepted way to determine and evaluate the correctness in searching using any of these systems and approaches, although observing the retrieval process and the estimations based on the designs of the systems can guide us in this.

The issue of evaluating information retrieval systems has become an integral part of the field. Since the first measures were developed in the 1950s, it has been a driving force in IR systems research and development. With several evaluation initiatives having been devised, it has gained considerable momentum over the years and the evaluation of retrieval of conventional information can be considered standard. Special attention needs to be paid to non-standard data like multi-lingual texts, geographical information, multimedia and indeed mathematics. With the mathematical retrieval systems only being developed in the last few years, the evaluation of this type of domain-specific retrieval is still in its infancy.

This thesis is a continuation of my Bachelor thesis, Vyhledávaní v matematickém textu [9], in which my math-aware searching system, Math Indexer and Searcher (MIaS), was designed and developed. The thesis looks at the history, initiatives and means of evaluating of information retrieval systems in general and tries to apply it to math-awareness of systems with the aim of devising suitable techniques to be used further in the thesis (Chapter 2 on the following page). It then proceeds with innovations in the world of mathematical retrieval since it was described in my bachelor thesis (Chapter 3 on page 21). It then proceeds to describe further development, design and fine-tuning issues as well as basic evaluation of the Web/MIaS system (Chapter 4 on page 24). The final chapter evaluates the system in accordance with the findings from the previous parts of the work (Chapter 5 on page 39).

2 Information Retrieval Evaluation

Information retrieval (IR) is a key technology to access corpora of information and to manage the knowledge contained in them. The first IR techniques were developed in the late 1950s and today such search systems are in everyday use by users. IR systems deal with analyzing, indexing and storing data and with retrieving information according to each user's current information need. To evaluate an IR system means examining its effectiveness, that is whether it satisfies the user's information need, as well as performance (speed and storage requirements) and usability.

2.1 History

The first IR evaluation questions arose shortly after their first design proposals. The first measures, recall and precision, were proposed by Kent et al. in 1955 and remain two basic measures used to date [21]. Later studies at Cranfield College of Aeronautics from the late 1950s to the mid 1960s then set the standard for IR evaluation which is used to this day, even though it was still pre-computer era.

Cranfield 1 subjected four different indexing schemes of how information should be organized [19] which provoked many discussions. These then led to the formulation of the Cranfield 2 Project which was a big step forward. One of the many outcomes was to use combinations of words for searching (which is the underlying principle of today's word-based search engines). Another achievement was the way queries and good answers for them were captured – for each document an author was asked to formulate questions, to which his paper was a good answer. This became the criterion for search engines. Methods of this project again became the source of arguments and discussions, but despite their limitations they survived in the field of IR and brought valuable results to it [19].

SMART (System for the Mechanical Analysis and Retrieval of Text) was an IR system developed at Cornell University in 1960s [20]. Many of the interesting and important concepts were developed with SMART research and nowadays form the basis of todays web-based search systems. SMART introduced the notion of scoring function as well as the ranking of retrieved documents based on it for displaying to the user. The system was based on the innovative vector space model[1] which is a multidimensional model. Is is used for representing text documents with each dimension corresponding to a different term in the document. The relevance of a term to a query is then computed as the distance measure in the space. SMART computed vectors not by a binary

1. Vector space model. Wikipedia. 4 Jan. 2013
`http://en.wikipedia.org/wiki/Vector_space_model`

occurrence of terms in the document, but by combining term frequency (TF) values with inverse document frequency (IDF) values. Another important step was an application of relevance feedback from users which improve current or subsequent searches [19].

The Medlars (Medical Literature Analysis and Retrieval Service) Demand Search Service – predecessor of today's Pubmed – was one of the first computer based IR systems [19]. It searched in medical literature. Interesting thing about this project were queries formulated only by expert searchers. Users communicated their information needs to experts by mail or face-to-face. Queries were run in batches at centralized server and the results were mailed back to the users afterwards. The evaluation of Medlars was also interesting. There was an effort for extensive analysis of failures in search – identifying relevant documents or other way around, rejecting non-relevant ones.

Karen Spärck Jones from Computer Laboratory at Cambridge also experimented in the IR field. She did not use her own test collection but used Cleverdon's instead. During her experiments, she invented inverse document frequency measure (IDF) which together with SMART's term frequency (TF) inspired future retrieval systems. Several deficiencies of Cleverdon's document collection occurred which led her to consider construction of suitable test collections for IR. Unfortunately this idea was not researched until recent involvement of TREC.

Later in the century, IR research started to focus on creating interactive user-based search – search that is accessible to end-user and is not mediated by search experts. Oddy and Belkin from the UK were involved in this research in the 1970s when the system called Thomas was developed. It maintained model of user interests. Oddy simulated real-user interaction and relevance judgments from other projects. In the 1980s the project Okapi started to focus on online access catalogs in libraries, which was the first sign of making computer-based search system available to practically anyone. It implemented a simple text search where queries were written into a box much like the ones today. Additionally, Okapi users were asked to give relevance feedback judgments for evaluation.

2.2 Initiatives

Research and development of IR systems, its methods and processes and its evaluation became inseparable. There are three major initiatives that are concerned in IR and its evaluation. Their purpose is to promote research, innovation and development as well as to provide needed infrastructure for evaluation of IR systems.

2.2.1 TREC

Text REtrieval Conference (TREC)[2] started in 1992 and it was the first large-scale information retrieval evaluation initiative. It is co-sponsored by the National Institute of Standards and Technology (NIST) and U.S. Department of Defense. TREC claims the effectiveness of participating systems approximately doubled in the first six years of the conference.

TREC consists of a set of tracks, i.e. areas of focus, in which particular retrieval tasks are defined. These tracks serve as an incubator for new research. The first run of a track defines what the problem actually is and it provides the infrastructure to support the research. Tracks also encourage robustness of retrieval technologies – same techniques can be used in more tracks. Tracks are also an invitation to a broader set of researchers interested in various fields of retrieval. Some of the tracks held over the years have been [11]: question answering, genomics, HARD (High Accuracy Retrieval from Documents), robust retrieval, spam track, blog track, terabyte, legal, enterprise and a million query track.

For each track, several tasks and the data used for them are agreed beforehand by the track coordinators as well as the track participants. The data consist of a collection of documents, a collection of search requests designated for these documents and a set of relevance judgments for request-document pairs. Documents and requests are distributed to participants which run their systems over the collection and post back the results afterwards. Participants then learn how they did with the challenges which is also a part of the conference itself.

One of the traditional tracks was the so-called ad-hoc track. It simulated a user who would sit down in front of a search system and would conduct a search against an existing collection of documents. The system returns a set of results (nowadays assumed to be ranked and ordered) that the user consults. For this model it is fundamental that the user be able to judge the relevance of each returned result irrespective of the order of the results presented [19].

Ranking algorithms, which are a crucial part in the behaviour of modern search engines, were also one of the main focuses of the ad-hoc task. Successful modifications of SMART's TF*IDF term presented at TREC helped to stimulate further developments in ranking.

Other interesting research includes relevance feedback (RF). With RF implemented the system learns characteristics of relevant documents only from the requests. It is not an easy task to evaluate systems using RF; the first trials were done by SMART. Okapi then used real users to evaluate RF ability – a user performed a search, examined several top documents and performed subsequent search. The user would then mark new documents that were relevant to their query that were not included in the first set of results. TREC introduced routing

2. Text REtrieval Conference. National Institute of Standards and Technology. 4 Jan. 2013 `http://trec.nist.gov/`

and adaptive filtering tasks, which were based on RF. These tasks helped to introduce machine learning practices into IR which over the years became increasingly common [19].

TREC also addressed the issues of searching the Web. There are several pitfalls in searching documents on the Web, the main one being the sheer amount of pages, their heterogeneity and quality variation. Very Large Collection track, Web Track and Terabyte Track tackled these issues. Linkage between pages has started to be used for document ranking, and Google introduced its PageRank – a query independent measure of the quality of web documents.

TREC's evaluative collections contain millions of documents. The first TREC collections consisted of newspaper articles. Collections for Terabyte Track was made by crawling the US .gov domain.

2.2.2 CLEF

Different languages often need different approach and optimization methods in retrieval. Conference and Labs of the Evaluation Forum[3] (CLEF, formerly known as Cross-Language Evaluation Forum) first took place in 2000. It followed TREC's model for evaluating, with the addition that it offered collections in several languages, namely English, French, Spanish, Italian, German, Dutch, Czech, Swedish, Russian, Finnish, Portuguese, Bulgarian and Hungarian.

Every year the topics are translated into several languages. Queries can be written in one language and topics in different languages may be retrieved, therefor queries need to be translated correctly to search within documents written in different language. Search results are returned by participants and are evaluated by native speakers of the tested language variations. Question-answering track uses a number of questions answered correctly as the main measure.

CLEF also consists of several content specific tracks: ImageCLEF (combined textual and graphic retrieval), iCLEF (interactive task track), domain specific track, WebCLEF, GeoCLEF (geographic retrieval), VideoCLEF.

2.2.3 NTCIR

NTCIR[4] stands for NII (Nation Institute for Informatics) Test Collection for IR Systems and it is a series of evaluation workshops designed to enhance research in Information Access (IA) technologies focusing on Asian languages. Its first workshop was held in 1999. Several tracks are being held: ad-hoc track, patent track, web and question answering track. Similarly to TREC and CLEF,

3. The CLEF Initiative. 4 Jan. 2013
`http://www.clef-initiative.eu/web/clef-initiative/home`
4. NTCIR Project. 4 Jan. 2013
`http://research.nii.ac.jp/ntcir/index-en.html`

collections for ad-hoc tasks consist of newspaper articles. The collection for the Patent track consists of 7 million documents, and there are 1,200 search topics that are made of refused patent claims. Web track collection comprises of approximately 1 TB of pages crawled from the .jp domain.

2.2.4 MIR workshop, CICM 2012

Conferences on Intelligent Computer Mathematics (CICM) in the year 2012[5] hosted a workshop focused on mathematics information retrieval called MIR 2012 Workshop[6]. It was the first ever officially organized MIR event, and it was conceived by assoc. prof. Petr Sojka (Masaryk University, Brno) and prof. Michael Kohlhase (Jacobs University, Bremen) at CICM 2011.

Among several talks on mathematics retrieval, the most important part of the workshop was MIR Happening – an informal friendly competition that aimed to show the abilities of contestant math search systems in real time in front of a live audience.

There were two different test collections both extracted from arXMLiv[7] project:

- sandbox – 10,000 full arXMLiv documents;
- harvest – 177 files each containing 10,000 non-trivial formulae extracted from arXMLiv documents.

These collections were injected with documents picked by three judges: prof. James Davenport (University of Bath), Dr. Patrick Ion (Mathematical Reviews) and Dr. Daniel Mayer (Jacobs University, Bremen). The injected documents were the correct results for judges' queries. The test document collection was published to contestants several days before the actual conference. Contestants had to index the documents and be prepared to query the system with judges' information needs.

MIR Happening consisted of three differently intended tasks:

- Formula Search – formula-only queries searched in the harvest test collection.
- Full-Text Search – text/formula queries searched in the sandbox collection.

5. CICM 2012. 4 Jan. 2013 `http://www.informatik.uni-bremen.de/cicm2012`
6. MIR 2012 Workshop. 4 Jan 2013 `http://www.cicm-conference.org/2012/cicm.php?event=mir&menu=general`
7. arXMLiv: Translating the arXiv to XML+MathML. The KWARC research group. 4 Jan. 2013 `http://kwarc.info/projects/arXMLiv/`

- Open Information Retrieval – natural language queries were left to the system operators into translate to appropriate queries to retrieve specific information.

Section 5.1 lists the queries prepared by the judges as they were released to participating "contestants". It was intended to measure precision and recall as well as some efficiency measures for the queries performed in the tasks.

Two systems, MathWebSearch and MIaS, joined the friendly competition. It turned out that the time dedicated for the Happening was not sufficient to perform all queries of all tasks nor to measure stated effectiveness and efficiency indicators. Trying to accomplish search tasks prepared by the judges was a great lesson and very interesting experience for the contestants, judges, organizers as well as the audience, as it invoked a good deal of discussion and interest. Overall, the workshop was considered very successful and a good starting point for the next event.

2.2.5 Math Task, NTCIR 2013

The 2013 NTCIR (see 2.2.3) conference[8] will host the first math search task[9] and it is the first time a conference focused on IR evaluation provides its facilities for this field of IR. The conference takes place in June 2013 in Tokyo, Japan.

NTCIR 2013 Math Task is divided into subtasks:

- Math Retrieval Subtask which is further divided, as it was at MIR 2012, into Formula Search, Full-Text Search and Open Information Retrieval tasks. A test document collection for this task consists of the initial 10,000 documents for a dry preparatory run and 100,000 documents for the official run.

- Math Understanding Subtask which aims at extracting natural language definitions of mathematical expressions. Document collection will consist of 10 annotated documents for a dry run, 20–40 for the official run.

The task has several deadlines scheduled several months in advance of the actual conference. Datasets were released in November 2012, search queries a month later. Participants had two weeks to post results back to organizers. Evaluation results are released 4 months before the conference for the participants to incorporate them into conference proceedings. The main organizes of the task are Akiko Aizawa (National Institute of Informatics, Japan), Michael Kohlhase (Jacobs University, Bremen), Iadh Ounis (University of Glasgow).

8. The 10th NTCIR Conference. 4 Jan. 2013
`http://research.nii.ac.jp/ntcir/ntcir-10/index.html`
9. NTCIR Pilot Task: Math Task. 4 Jan. 2013
`http://ntcir-math.nii.ac.jp/`

2.3 Evaluation in IR Systems

2.3.1 Collection

To measure the effectiveness of an information retrieval system in a standard way, a test collection that is composed of three components is needed:

- Set of documents. A document collection must be representative for the application in size and the type of documents. It can be the one a system normally uses or it can be a sample from it or a similar collection.
- Set of information needs – queries. Queries should also be representative for the particular application. These can be gathered from the actual logs of what users search for, or they can be collected by interviewing potential system users for examples. The size of the set of queries is also very important. It can incorrectly resolve between correct and incorrect behaviour especially when evaluating two or more systems [4]. In some applications a great number of test queries can be gathered, but to establish relevance judgments for all of them imposes a constraint.
- Set of relevance judgments for pairs of documents and information needs. These can be done by the users who are evaluating the system or by a third party judges who have been properly instructed. In a binary system there are only two possible relevance judgments: relevant and non-relevant. If a system is evaluated by actual users a three-way system can be proposed: relevant, possibly relevant and non-relevant. Possibly relevant documents can be then assigned to either of the two remaining groups depending on what effectiveness measures are used [4].

Basic approach to IR system effectiveness evaluation is built on a notion of relevant and non-relevant documents. A document is relevant to an information need if its content addresses the given information need. The documents in the test collection are given binary judgments as to their being relevant or non-relevant with respect to this query. This decision is referred to as the gold standard or ground truth judgment of relevance [12]. Making these judgments for every document-query pair consumes a lot of time and resources. Cranfield experiments tried to complete exhaustive relevance judgments, while later experiments judged only a subset of documents. The test collection and the set of queries need to be a reasonable size to average the performance of a system, despite it being more challenging to complete relevance judgments for larger sets of documents and queries.

In a binary system a document is considered either relevant or non-relevant to a query. Usually modern IR systems are ranked – some documents can be highly relevant, others only slightly, some not at all. These systems use various weights, factors and formulas to compute the level of relevance of a document

to a certain information need. Results returned by these systems are ordered according to their scores. To evaluate the correctness of a ranked result set is naturally even more difficult than binary relevance statements.

When relevance judgments need to be created for a relatively large collection usually only the top k documents retrieved are judged (for TREC, k is between 50 and 200 [4]). This method is also called pooling. Top k results from more than one IR system are collected and unified into a pool which is manually judged for relevance. A new system or an IR algorithm which did not contribute to the original pool can however have problems with being evaluated correctly if it returns completely different results than the original pool creator systems.

One valuable source of relevance judgments can be the logs of users' actions. Such a log usually consists of a complete query a user posted to a search system, a list of top results that were returned and most importantly, records of which items from the result list were clicked on [4]. When a snippet of the document that matched a user's query is shown together with a link to the actual document, a user is able to evaluate the relevance of individual results based on the snippet and chooses the most relevant one by clicking on it. This approach obviously can not be as precise as regular relevance judgments, since some clicks can be false positives. This can, however, be an interesting insight into what a user finds relevant to their query.

2.3.2 Measures

Two basic measures for IR system evaluation with binary relevance classification are precision and recall.

- Precision P is the fraction of retrieved documents that are relevant:

$$P = \frac{(\text{relevant documents retrieved})}{(\text{retrieved documents})}$$

- Recall R is the fraction of relevant documents that are retrieved:

$$R = \frac{(\text{relevant documents retrieved})}{(\text{relevant documents})}$$

The basic requirement is to have a system with very high precision and very high recall, which is very difficult to achieve. It is very easy to obtain recall 1 – the system returns all documents for every query which also means that all relevant documents are always returned. This would, on the other hand, lead to an extremely low precision.

Precision and recall measures can be differently important in different environments and situations. For example, web searchers demand high precision results, as they expect most of the results on the first or first several pages

to be relevant as they have no intention of looking through all of the results. Conversely, some professional searchers demand high recall as they want to retrieve most of the relevant documents in the collection and are willing to go through the possibly all of the returned results.

The measure that combines precision and recall is called F-measure [18]. The general formula for the F-measure is:

$$F = (1 + \beta^2)\frac{P \times R}{\beta^2 P + R}$$

Default balanced F-measure weighs precision and recall the same and is therefore computed as

$$F = 2 \times \frac{P \times R}{P + R}.$$

In modern search applications, users tend to look only at a certain number of top listed documents retrieved by an IR system. This is considered in evaluation of search systems and it also comes in handy when there is a huge amount of documents in test collections, where exhaustive relevance judgments is not feasible. Recall-precision measures are calculated only for the top k documents (usually 10 or 20).

Most of the IR systems produce ranked results. One way to measure a ranked result set is to compute the recall and precision measures for every rank position. By examining several top results, measured numbers can be displayed in a recall-precision graph. The graph usually has a sawtooth shape – if the $(k+1)^{th}$ retrieved document is not relevant the recall stays the same but the precision increases; if it is relevant both recall and precision increase.

An enhanced measurement to the one described above requires to calculate the precision at fixed recall levels from 0 to 1, usually at 0.1 steps. This method has the advantage that the effectiveness of the system is computed for all relevant documents in the collection, not just the top k ones. It is difficult to get these numbers from a basic recall-precision graph where numbers are not calculated at standard recall levels. To get the recall-precision measures for every recall level, interpolation is needed. Interpolated precision P at a recall level R is defined as the maximum precision measured at any recall level higher than R:

$$P = \max P(R')_{R' \leq R}$$

Interpolation removes the jiggles from the graph which consequently develops a monotonically decreasing tendency.

Another method is to average the precision numbers of the ranks at which relevant documents were found. The result is a single number which is based on the rankings of all the relevant documents. This metric values as many relevant documents to be returned as possible but at the same time it places weight on highly ranked relevant documents [4].

Measuring recall-precision numbers for the top k documents is unfeasible when there are many documents relevant to a query or when relevant documents are spread across the ranking. One way to measure in such a situation is to compute recall and precision for only a small number of certain rank positions. When comparing two or more result sets for one query it is sufficient to calculate precision only – recall has the same tendency as precision compared to another ranking. This method is called precision at p and it does not distinguish differences between rankings at positions 1 to p. With this method a search task focuses on finding most of the relevant documents at a given rank rather than finding all relevant documents.

In some applications it is important to return relevant documents as close as possible to the top rank, especially when users tend to look only at the top few result entries. One measure useful in this situation is the precision at p mentioned above. Another one is the reciprocal rank. This is useful mostly when there is a single relevant document for a query. It is an inverse number of the rank at which the relevant document was returned. It is more usual for the mean reciprocal rank based on more queries to be computed [4]:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}.$$

Another method which is called discounted cumulative gain is based on graded relevance judgments. Grades gain several values from "bad" to "perfect" and are assigned to several top rank positions. Discounted cumulative gain at rank position p is computed as:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

where rel_i is the graded relevance for a document at rank position i.

As mentioned above, an IR system is evaluated using not one but many queries. When we have calculated the average precision for every query, we can also compute arithmetic average of these numbers to get the final figure. It is called mean average precision. It is a single number that summarizes the effectiveness of a search engine. While it is easily readable, in some applications it provides little information and can not replace the recall-precision tables or graphs that it is based on.

Described above are measures for evaluating the effectiveness of IR systems which assesses how well the system works in finding relevant documents in a collection and in not returning non-relevant ones. There is also a performance side of the evaluation – efficiency evaluation. Efficiency evaluation can only be performed after the system has been set up to work in certain recall-precision levels.

Efficiency is measured by several indicators. Queries per second, or throughput, is a single-number metric that can help users in planning their query load but also indicates to system administrators whether the system runs on a sufficiently powerful hardware. Another measure is elapsed indexing time. It is the amount of time needed to create a complete index of some collection with a certain system setup. Slightly different metric is CPU indexing time, which is the amount of processing time needed to complete the indexing of some documents irrespective of parallelism, if used. Another IR efficiency metric is query delay. It is the amount of time that elapses between sending a user query and displaying the results back to the user. The last of the most commonly used metrics is index size. It is a space occupied by a document index of some collection. Usually, there is an effort to minimize the index footprint.

Similarly to the recall and precision, we demand systems to have a high throughput but very low latency. In a real system, throughput is not a variable but a requirement. There are two variables: latency and hardware cost [4].

2.4 Math-Awareness Evaluation

The previous section provided an overview of the most common techniques for evaluating the standard IR systems that have been developed over several decades. We learned they are quite versatile and in their general form can be used to evaluate possibly any type of retrieval system. In image, video or for example audio retrieval, one can still judge the relevance of one media sample to another sample and therefore perform any type of evaluation based on recall and precision and their derived metrics as described in the previous section. The main question then becomes one of judging the relevance of a piece to some whole, to be able to capture the ground truth so it can be assumed in further evaluation.

With mathematics, however, this is quite challenging to accomplish. Let us compare a math search to a conventional text search. Imagine a user searching for the keyword *dog*. The system returns a set of results. It is quite easy to evaluate the relevance of each result to the query. We can, for example, search the document linearly word-by-word to see, whether the returned document contains the query keyword. A more complicated example would be synonym search. When a user searches for a word, say *beautiful*, in a synonym search the system can also find documents containing words *wonderful, stunning, lovely, nice*, etc. The original word does not need to be contained in the document. Instead, for each word, the system uses a dictionary of words and its synonyms and when a user posts a query, the system expands the word in the query and adds its synonyms to it for the system to be able to search for any word from the list of synonyms. To evaluate such a search a similar dictionary is necessary, ideally a perfect one with all possible synonyms for each word in test queries.

We could then again search word-by-word for any of the synonyms to a query keyword and establish relevance judgments for the set of results.

Now let's try to search mathematics and evaluate the results. Let us say that a math-aware system is queried for a simple expression $a + b$. A system, depending on its math processing algorithms, could return only documents with occurrences of the exact same expression but also an expression with different variables $x + y$, which is semantically the same. Other system, however, could return results containing exact matches and matches with varied operators $a - (-b)$, which is also mathematically equivalent. Systems work differently because of their different architecture and math preprocessing techniques. To evaluate these systems, a ground truth needs to be established – a relevance judgment for documents in the collection with respect to the query $a + b$. To complete exhaustive relevance judgments we need to check every formula in all the documents and verify its mathematical equality against the query. This can not be done in the same way as it can in regular IR evaluations.

In a document, a formula is analogous to a sentence in natural language. Both consist of smaller tokens – words in a sentence, subformulae in a formula. Analogically to a text search in which single words are searchable, we want to be able to search for subtrees of a formula right down to atomic entities, such as identifiers, operators and numbers. When querying a system with formula $a + b$, we could expect documents containing this formula as a part of some other formula to be returned. For example a document with fraction $\frac{1}{a+b}$ could be relevant to our information need. This further complicates the evaluation.

A document D is relevant to a query Q if any of the subformulae $f_1, f_2, f_3, \ldots, f_n$ contained in the document D is mathematically equal to any of $q_1, q_2, q_3, \ldots, q_n$ formulae contained in the query Q:

$$D_{relQ}\{f \in D | q \in Q | f \equiv q\}$$

Based on this assumption, the relevance of a document to a query formula can be judged. In our example, documents containing either formula $a - (-b)$ or $x + y$, but also, for example, $a + 2b - b$, as well as an indefinite number of other expressions that are mathematically equal to the query formula $a + b$, would be considered relevant. This is a very challenging task to do.

It is near to impossible for any human, no matter how mathematically proficient, to produce a reasonable amount of relevance judgments for math query–math-contained document pairs. A human approach is suitable for creating only very small test collections. The best way possible to accomplish accurate relevance judgments would be to employ a computer algebra system. Such a system, among many other capabilities, is able to decide for any two formulae whether they are equal or not. Since it is a computer program, it would probably be possible to automate it in order to create an evaluative collection. However, this approach does not support finding matching subformulae natively, therefore extraction of all subformulae from all document's formulae

would be needed. Compared to manual human elaboration this approach is still suitable.

There are several capable computer algebra systems. One of the best known is Maple[10]. Among a great deal of applications it is also used for evaluating user answers against preset correct answers in math tests. Another system is Wolfram Mathematica[11] which too has a vast range of applications.

To help us determine whether a query formula or it's similar form is contained in a document we can make use of other systems computing similarity of objects. They work mostly heuristically and are usually focused only on specific form of similarity. As a matter of fact, every math-search engine is based on such a system – MIR systems need to know whether a formula in some form is contained within a document. In more general, this naturally holds also for any IR system. For instance, in MIaS it is an M-term (see 4.2.7) computing facility. These methods are in essence the ones we need to evaluate and they are usually limited, so it cannot be used for setting the ground truth, however, using one system's formula-similarity computations could in some cases help us in evaluating other systems.

Another approach could use a system that calculates the similarity of images (Google Images[12], Mufin[13]) of rendered formulae. Of course, this could in no way achieve full mathematical equality but the similarity based on a formula and subformula structure could be evaluated. This raises a question of employing such an approach in a MIR system itself. The same naturally applies for employing computer algebra and other systems which can decide if two formulae are equal or not or at least similar.

Described above are some possible methods for automatically building a test corpus for a math aware IR system. It could lead to extensively evaluated relevance judgments for every query-document pair. Especially when used in co-operation with computer algebra system, we could probably declare it as fully mathematically correct. This is similar to Cranfield's ambition to create extensively evaluated test corpora, which turned out to be deprecated. That also applies to math evaluated test collection.

Extensive relevance judgments supporting full mathematical equality would unquestionably test math IR system application's accuracy. In some cases full mathematical equality in a search system and its evaluation is not needed. Say a user wants to find the exact occurrences of the expression $16/2$. In such a case, they does not want all the occurrences of the number 8 to be returned.

There are several use cases in which math search can be useful:

10. Maple. Maplesoft. 4 Jan.2013 `http://www.maplesoft.com/products/maple/`
11. Wolfram Mathematica. Wolfram. 4 Jan. 2013 `http://www.wolfram.com/mathematica/`
12. Google Images. Google. 4 Jan. 2013 `http://images.google.com/`
13. The MUFIN Project. 4 Jan. 2013 `http://mufin.fi.muni.cz/tiki-index.php`

1. to find out in which context and applications a formula is used,
2. to recall a document on a piece of expression,
3. to find out whether and where a similar construction has already been used,
4. to specify and narrow down text search results,
5. to simply put in a formula and see what a system finds.

These application depend on the type of a user currently using the system. It can be a professional mathematician, professionals in fields such as physics, engineering, biology, a student of mathematics or a completely random non-math user, which is however not very likely. The design of math search systems should consider these most probable use scenarios and so should its evaluation. Otherwise the system could be evaluated perfectly suited to applications that are never actually used and therefore unusable.

2.4.1 Devising a Test Collection

There are several ways to construct a test collection:

- A set of documents is selected, a judge reads over them and then creates several queries that he knows should be answered by some of the documents. This approach is similar to basic approach in regular IR, developed by Cranfield. It consumes a lot of resources and never can produce a large test collection.
- A set of documents is selected and a judge injects it with documents that has been already established to be relevant to questions the judge has also prepared. This approach is better for constructing larger collections, but it can never be known whether there are any other documents in the collection that should also be returned. The system should return the specific document that this query asked for among its top results.
- A set of documents and queries are prepared independently. Top k results need to be evaluated by a judge for false positives. False negatives will not be detected.
- A set of queries is evaluated against a set of documents using a computer algebra system. As described in the previous section the parameters of an evaluating system need to be carefully set up for relevance judgments to be useful for a particular application.

These methods in some way correspond to math search use cases stated in the previous section. For instance, if we want to recall a document on a piece of a formula that we already have, we want this document to be among the top results and ignore the others. Evaluations based on injected document seem appropriate. If we would like to see only what a system returns for a query, an exhaustive evaluation matches this case.

Test collection is an essential part in evaluation of information retrieval systems. While in regular IR it is possible for a non-specialist in the topic, even a system developer, to build such a collection, in mathematics IR it is quite important to have a mathematician's knowledge to help to create the collection. The knowledge in the field of mathematics can substantially help to improve much needed semantic orientation of the test suite and consequently influence the semantic orientation of the search systems themselves. A non-mathematician, usually a system developer, primarily focuses on a structural and similarity correctness of matched formulae and does not place much importance on the true meaning of the query and its expected results. While this is not necessarily a bad approach, in MIR it is only one of the use cases.

2.4.2 Measures

An important issue when evaluating a MIR system concerns the effectiveness indicators to be measured. For better usability MIR systems should return ranked results which is one of the parameters for the measures used. They are also dependent on the type of test collection we have prepared following possible creation techniques mentioned in the previous section which are further linked to the possible use cases. If an exhaustively judged collection is available, recall-precision and their derived measures can be computed for an arbitrary number of top documents. If we are limited to relevance judgments for only several top documents, we can use measures that make use of this limitation, precision at p being one of them. However, if we have a test collection with injected to-be-returned documents, most probably only one for a query, it is possible to make use of reciprocal rank which scores system's performance in returning the only relevant document there is for a query. Otherwise, recall-precision numbers, F-measure, average precision at certain ranks and precision at p can be calculated for top k results. Generally in IR evaluation and especially in MIR the lower the k the less power-consuming and less difficult the evaluation.

Measuring and evaluating efficiency is also important in the IR, especially with MIR where an additional complexity in the processing methods of mathematics can place an extensive load on some of the system resources. During the indexing of the document base, CPU can be loaded heavily which together with hard drive speed, the amount and speed of RAM memory can significantly affect the final indexing time. It is therefore important to measure it. After indexing there can be a lot of tokens in the index stored, especially if the system

expands formulae from the documents and stores variations of them in the index which puts a high demand on the storage space. A system that is not based on a full-text core can require a lot of RAM to function properly when searching. Hence it is also important to be aware of performance figures as they can affect the scalability of the whole search solution both in terms of the storage space as well as the query time.

3 Recent Advances in Math Retrieval

My bachelor thesis in 2010 [9] explored different efforts in developing MIR systems that were ongoing at that time. Methods, data structures, data formats and querying possibilities were described, compared and finally summarized in one table, which was subsequently extended with my approach, the new MIaS system [23]. This chapter aims at exploring further developments in the mathematical information retrieval world since its last summary. Changes in previously stated systems as well as a system, which was not mentioned in [9] are described here.

3.1 EgoMath v2

The main focus of the EgoMath system is on enabling math searching in semantically poor, real-world documents by supporting similarity searches. It is enabled mainly by two algorithms – augmentation which produces several more or less generalized representations for one formula, and an ordering algorithm which finally orders operands in augmented formulae to get canonical representations. Expressions are indexed like normal words in TEX-like postfix notation to avoid using parenthesis. EgoMath version 2 was released in 2011. The new version is based on the reimplementation of most of the system components [14]. The new implementation offers new features such as the ranking of results which is very important. Configuration for math processing algorithms was moved to separate XML files which allows the system to have its behaviour quickly adjusted for different document collections. The web interface was also reimplemented and is now able to display snippets showing matched formula representations in both its original and its derived form. EgoMath v2 focuses on being a complete package and aims for a simpler administration of the system and the indexes through its web UI. The new system is now able to perform indexing several times faster than its predecessor [14].

To demonstrate Egomath's orientation on searching web-based math documents an attempt to index math pages from Wikipedia.org was made. More than 28,000 math articles were extracted with about 240,000 mathematical elements [14]. TEX formulae were converted to MathML by LATEXML converter[1].

Unfortunately the published web interface[2] is down at the moment which raises questions about the project's future development.

1. LaTeXML: A LaTeX to XML Converter. 4 Jan. 2013 `http://dlmf.nist.gov/LaTeXML/`
2. EgoMath. 4 Jan. 2013 `http://egomath.cythres.cz:8080/egomath/`

3.2 DLMF Search

Digital Library of Mathematical Functions[3] (DLMF) is a web counterpart to the NIST Handbook of Mathematical Functions, a project of the National Institute of Standards and Technology (NIST) [15]. Its goal is to provide a reference for researchers and other users in applied mathematics and many other research fields who daily encounter special functions in the course of their work.

Such a portal needs a math search functionality to help users to manage the amount of mathematical knowledge served. Despite the development of the DLMF search since 2006 it was not covered in [9]. It has been designed to follow several general but important goals [28]. To allow a similarity search, a query relaxation is used. It modifies and adds new similar query terms to an existing query. Query relaxation also covers basic forms of equivalence in terms of commutativity and associativity. The system ranks relevance of matched terms based on their nature – definitions rank the highest, followed by theorems, function names rank higher than variable names etc. The system indexes enriched metadata of equations and expressions so that it can search for these objects by their natural language expression. Hit highlighting is also supported to give the user a quick clarification of why the document was matched. Fragments for displaying to the user are stored in the secondary index for a better search time response. Fragment index is queried in search time to retrieve appropriate excerpts for the query and fine-grained elements are highlighted [13].

The system follows general math-aware search system design and feature suggestions, however, the DLMF search system is serving a closed environment with predictable data and metadata. It would therefore be interesting to see what modifications it would need in order to index and search real world documents. The user interface of the system seems functional and sufficient. There are few options to search for a page, equation, figure or a table.

3.3 MathWebSearch

As summarized in [9] MathWebSearch is a system that differentiates itself from other approaches by not being based on full-text indexing engine and by focusing primarily on semantics of formulae contained in the documents by making use of their semantic markup. This can be a problem since most of today's papers do not contain semantically enriched formulae. However MathWebSearch indexes documents from arXMLiv [25] project which heuristically converts LaTeX encoded documents to combined presentation-content MathML.

MathWebSearch uses substitution trees to index mathematical expressions. Substitution tree is a tree-like structure where the root node is a generic term

3. Digital Library of Mathematical Functions. NIST. 4 Jan. 2013 `http://dlmf.nist.gov/`

and every child node represents one or more substitutions of one of the terms [7]. By the nature of this solution and with the help of four types of queries MathWebSearch introduces: instantiation queries, generalization queries, unification queries and variation queries, and it also supports the equivalence of expressions up to α-equivalence [7].

The new version, MathWebSearch 0.5, introduces changes in how substitutions are applied in the substitution trees. Every node in the tree now represents one substitution that is always applied to the left-most free variable which simplifies the lookup of the right substitution [8]. New version also brings RESTful public HTTP API which decouples the server and the client side of the application which communicate through *mws*-namespaced XML structures [16].

MathWebSearch developers have also started to deal with evaluation, however only the efficiency of the system is measured. The system is evaluated using arXMLiv collection of documents that were converted with no errors containing ca. 115 million expressions. Query times and memory usage have been reported as performance indicators with respect to the document collection size [8]. Based on the evaluation of the memory requirements of MathWebSearch, main focus of the developers now became creating a distributed system.

3.4 Summary

There is not much development going on in the mathematical area of IR. Only a few projects are kept alive, mainly those that are employed on narrowly focused portals within a closed data environment such as LeActiveMath and DLMF search. Other systems' main development focus is on increasing precision and accuracy and increasing applicability by developing a better user and inter-machine interfaces. This is mainly because the number of possible real applications for the MIR system is much smaller than for the text search that is used by millions of people every day. Another reason may be the lack of an evaluation framework and background for this type of search which suppresses competitive development for the benefit of users. This is expected to improve in the near future with math search evaluation workshops starting to occur (cf. Sections 2.2.4 and 2.2.5).

Math Indexer and Searcher system has been also evolving in many areas as is described in detail in the next chapter.

4 Development of the MIaS System

Math Indexer and Searcher (MIaS) is a math-aware information retrieval system that was developed and first described as a part of my bachelor thesis [9] called Vyhledávání v matematickém textu (Searching Mathematical Texts). It researched existing approaches to math searching to date and based on the findings designed and developed its own solution for real world usage. It aimed to be used in environments with large document bases as those in digital mathematic libraries such as EuDML [26] or DML-CZ. Math processing functionality used in MIaS was subsequently integrated into EuDML's search system[1].

The system as described and provided in [9] was in its very first version, and further development was needed to accomplish the stated goals. Some of the work done was partially described in [23, 10, 24] together with proposals of future work, only some of which was undertaken. This chapter aims at bringing all the development since the system's first description to one place and describes the changes and new features in appropriate depth. This chapter also invoked further mainly implementational changes to the system in the refactoring area, the goal of which along with functional improvements is to create a new version of the system.

4.1 System Summary

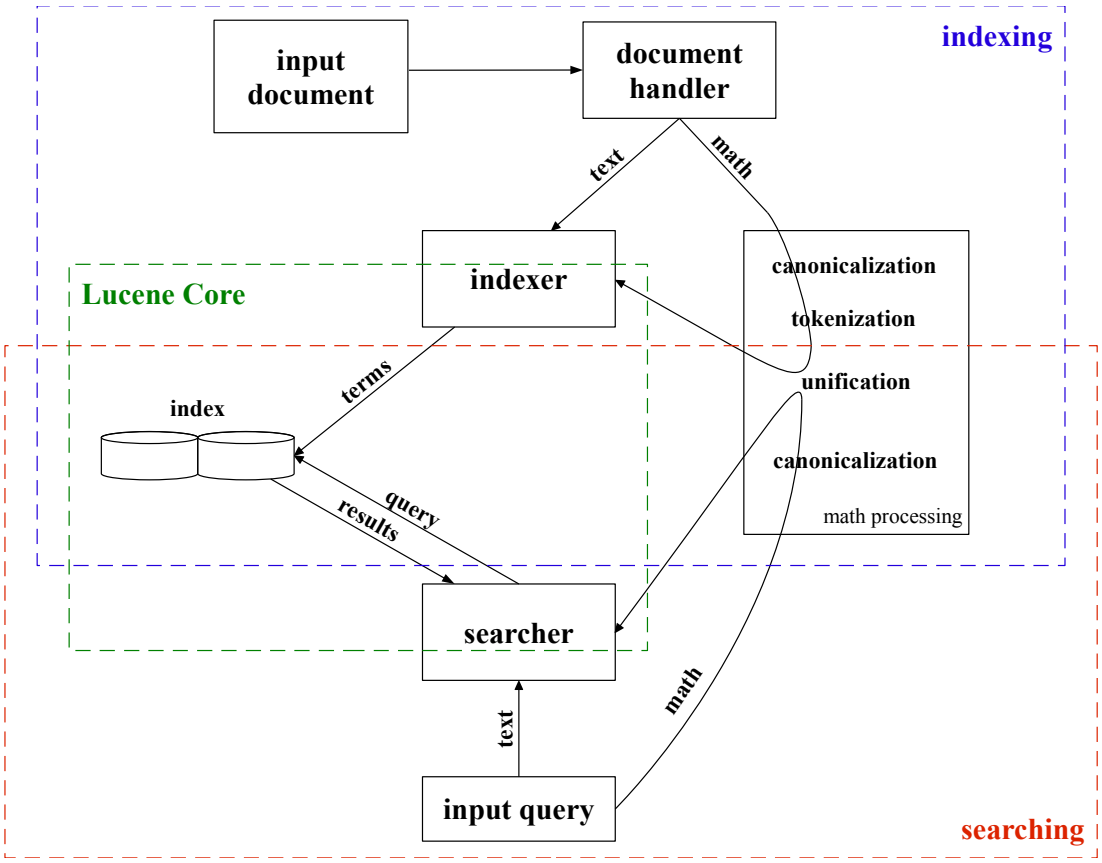
MIaS is a math-aware full-text based search system. It enables users to search for mathematical formulae and expressions contained within indexed documents encoded in the MathML [3] format. It is a Java-based server application and is coupled with a web interface, WebMIaS. This eases the use of the system and is able to bring functionality of the MIaS to the wide public. The application was built on top of the state-of-the-art full-text indexing system Lucene[2] (current version 3.6.2). Mathematical preprocessing part (4.1) is built as an extension and can be easily plugged into other Lucene or Solr based systems and with few modifications possibly to any full-text search engine.

The general overview of the system's components and its workflow is displayed in Figure 4.1. The operation of the system can be split into two phases. Offline indexing of files is started by assigning an appropriate document handler. Handlers treat different types of documents differently according to the need for the collection. Indexing continues by preprocessing mathematics and analyzing text contained in a document and finishes by storing all generated

1. Advanced Search. EuDML. 4 Jan. 2013 `http://eudml.org/search.action`
2. Apache Lucene. The Apache Software Foundation. 4 Jan. 2013 `http://lucene.apache.org/core/`

mathematical and textual tokens in an index. In the second phase the system is queried in real time by users. This phase handles user input very similarly to the indexing phase, generates math and text query tokens which are connected to a final query with which the system searches the index.

Figure 4.1: Overview of the MIaS system components and workflow



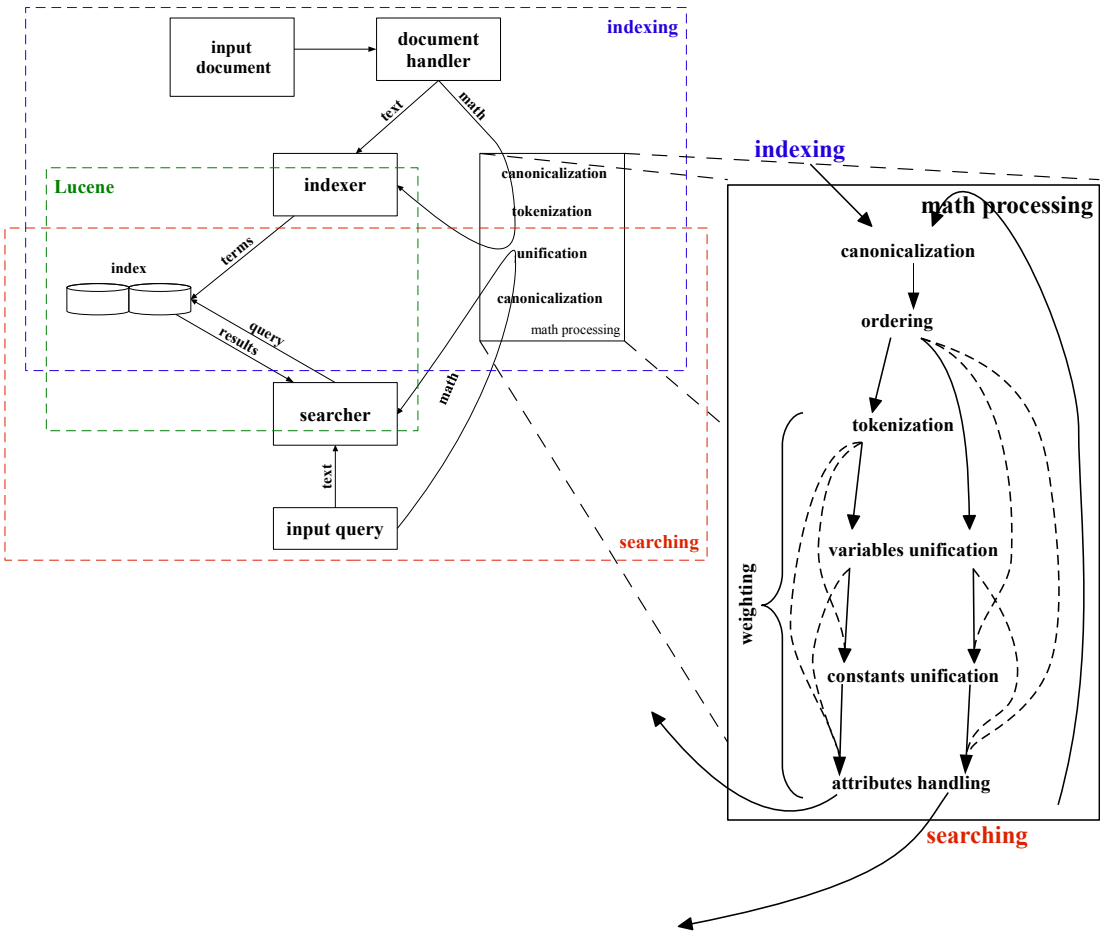
Most of the design decisions and implementation details are described in [10]i. However, the class and the package structure has changed rapidly. The whole application was moved towards being more object oriented. For a better understanding of the program's classes and their interactions, refer to the class diagrams (Figures B.2 and B.3) and the component diagram (Figure B.1) in Appendix B on page 54.

4.2 Math Processing

Special attention is being paid to the processing of math formulae so as to enable similarity searches – users are able to retrieve not only exact matches for their queries but also equal formulae written differently up to a certain level of mathematical equivalence, similar formulae and subformulae. The

system is full-text based, therefore handling all sorts of problems with matching mathematical expressions needs to be underpinned in the preprocessing stage. The matching of tokens is then left to well-established conventional full-text search. Expressions are processed in several steps, functions. Every function creates modified formulae which are more generalized than the original one. Every formula is assigned a weight which denotes its distance from the original form extracted from a document. This weight enables the system to rank the results and order hits accordingly from the most relevant to the least. A detailed overview of the preprocessing of math is displayed in Figure 4.2. There are five methods currently carried out in the math preprocessing module: canonicalization, ordering, unification of variables, unification of constants and finally MathML attributes handling.

Figure 4.2: Math preprocessing



4.2.1 Canonicalization

Documents containing mathematics can originate from different sources, i.e. generated by different LaTeX (and possibly other formats) to MathML converters.

The notation can therefore vary slightly depending on the converter used. MIaS converts MathML expressions into string tokens which means that any changes in notation will cause incorrect behaviour. Canonicalization turned out to be a very important preprocessing method in the workflow. Ideal canonicalizer would be able to convert different, but semantically eqivalent MathML formulae generated from different sources to one canonical representation. This is crucial when a MIR system aims at indexing documents from different sources. Additionally, there is usually a converter on the user input side of the system to ease the input of queries; at the same time a system might by able to accept copy-pasted MathML. All these indexing and querying inputs need to be canonicalized to one representation which is then handled by the core. MIaS has so far indexed documents with MathML generated by LaTeXML but users of WebMIaS are able to input LaTeX queries which are converted by a converter. As well, they can directly paste MathML. Experiments with using the UMCL library [1] as a canonicalizer turned out to be unsuccessful and showed that it is not a simple task. This led a team at Masaryk University to develop its own solution mainly for the use of MIaS [5]. Description of the canonicalization tool is beyond the scope of this work. It is important to apply the same canonicalization on input documents during indexing as well as on user input in the searching phase.

4.2.2 Ordering

An ordering algorithm is a form of unifying two formulae with the same structure but with permuted operands. The simplest example where search engines can benefit from this type of method is an expression $5 + x$ searchable by a query $x + 5$. MIaS currently orders operands of the commutative operations, addition and multiplication. The method traverses the structure bottom-up and uses the names of MathML elements that denote the operands to order them alphabetically. If two operands have the same element name, linear strings of the already sorted subtrees is constructed and compared for order.

4.2.3 Tokenization

When a user searches for a formula he expects to find not only an exact whole match but also formulae which contain his query as a part. Consider a query $\frac{1}{x^2}$ for which an appropriate result can be $\int \frac{1}{x^2} dx$. Tokenization is a method in MIaS math preprocessing which extracts subformulae from a formula. It traverses the formula's tree depth-first down to single variables, numbers and operators. All retrieved expressions are added to a list and passed for further preprocessing. A configuration parameter here is the maximum depth which the system retrieves sub-expressions from. There can be a decision that says that expressions retrieved from, for example, the tenth level and deeper are barely relevant to the whole document and for this reason, such expressions do

not need to be indexed. Such configurations can have a noticeable impact on the system's scalability in terms of index size and indexing time. MIaS currently indexes all expressions from all depth levels, including single variables and numbers, excluding single operators.

4.2.4 Unification Methods

To enable searching for similar expressions there are currently two formula unification methods used. The unification of variables is a substitution of all variables (values of MathML element `<mi>`) for unified symbols. A simple example of the benefits of this method is the query $a + b$ that matches the expression $x + y$ in the index. Unification of variables takes bound variables into account which means α-equivalence is provided. A downside of this approach is that the `<mi>` MathML element denotes not only variables, but generally any identifier that can be found in a formula – constant names, function identifiers and other special symbols that can have established meaning in the field. Stand-alone identifiers (expressions with one identifier element) are not unified.

Similar to variables, numbers are also unified in MIaS by simply substituting them for one unified symbol. This allows the system to match expressions with different numbers, which can be handy when a user does not remember an exact number or, for example, makes a mistake in the decimal part. In some cases, however, this can be too much of a generalization. Stand-alone numbers are as well not unified.

4.2.5 Attributes Handling

Attributes of some MathML elements can considerably affect the semantics of formulae. In Presentation MathML it is, for example, a `mathvariant` attribute, which defines the font style for an element. MIaS therefore indexes expression in both versions – with and without such attributes, allowing matches with matching attributes to rank higher.

4.2.6 Weighting and Fine Tuning

Ranking hits is one of the most important aspects in the search process. Hit ranking designates the order of search results and can have considerable a affect on the perceived quality of the system. MIaS depends on a fine tuned hit scoring function of a conventional text search (i.e. TF-IDF) and adds a math-specific parameter to it and argues with [29].

The methods described in Section 4.2 produce formulae that are more generalized than their original forms. MIaS indexes all versions of math expression that are generated in the process, which also means that all these expressions can be matched when searching. The system needs to discriminate between

these matches based on their generalization level. More generalized formulae are assigned a lower weight so they appear lower in the results than less generalized formulae hits. There needs to be a metric which defines the distance of a formula from its unmodified untokenized form.

Every formula that is processed by MIaS holds a weight value which defines the distance from its original formula. It is a number within the interval $\langle 0, 1 \rangle$ and since the ordering method does not lose any information from the formula, the original ordered representation has a weight of 1. Math preprocessing methods that generalize formulae in some way (tokenization, unifications, attributes handling) use a factor by which they multiply the weight of an input expression and assigns the new weight to a newly produced representation.

Devising the values for these factors is not a trivial task and there is probably no generally suitable set of values. They are and should be dependent on the type of documents and their field of study. The value of the numbers in formulae in applied physics can, for example, be very different from theoretical mathematics.

In MIaS, values of these coefficients are static across all different possible document bases since there has been no need to make them adjustable so far. However, making them so for indexing different corpora is easily achievable. Current values have been determined by an empirical approach with the help of a tool that was specially developed for this purpose (see Table 4.1 on the next page). In this tool several inputs can be specified: a query formula, several formulae to be indexed and searched in as well as values for every factor in math weighting function. As a result, weights of indexed formulae and final scores of hits can be inspected.

Several considerations came out from the experiments with different weight coefficients and these differ from the original ideas in [9]. The factor used by the tokenization method (Section 4.2.3) which is applied when a subformula from the next level is extracted still remains at 0.5 as well as the coefficient used by the unification of variables (Section 4.2.4) which is 0.8. The main differences from the original model is that MIaS now considers unification of numbers more information-loosing than variables unification; its coefficient has a value 0.6. This means that an exact match is more relevant than a match of an expression from one level higher with both variables and numbers unified, but is still less relevant than expressions with only one unification method applied. Also MIaS now places a strong emphasis on the complexity of the original formula in which a match was found. The idea is that a match in the expression that is more complex, i.e. has more elements, is less relevant than in a formula with fewer elements. The difference from the original model, where every formula in its original form was assigned a starting weight 1, is that the starting weight is an inverse number of the square root of the number of nodes that the formula consists of [23]. This causes overall weights of indexed expressions to be considerably smaller, but it does not affect inter-formula comparisons

Table 4.1: Example of application of weighting function on several formulae. The query is $a+3$ – all queried expressions are $a+3$, id_1+3, $a+\mathrm{const}$, $\mathrm{id}_1+\mathrm{const}$.

Formula	Indexed Expressions	Score	Matched
$a+3$	0.25=[$a+3$], 0.2=[id_1+3], 0.175=[a, 3, +], 0.125=[$a+\mathrm{const}$], 0.1=[$\mathrm{id}_1+\mathrm{const}$]	2.7	0.1[$\mathrm{id}_1+\mathrm{const}$] + 0.25[$a+3$] + 0.2[$\mathrm{id}_1+3$] + 0.125[$a+\mathrm{const}$]
$b+3$	0.25=[$b+3$], 0.2=[id_1+3], 0.175=[b, +, 3], 0.125=[$b+\mathrm{const}$], 0.1=[$\mathrm{id}_1+\mathrm{const}$]	1.2	0.1[$\mathrm{id}_1+\mathrm{const}$] + 0.2[$\mathrm{id}_1+3$]
$a+5$	0.25=[$a+5$], 0.2=[id_1+5], 0.175=[a, +, 5], 0.125=[$a+\mathrm{const}$], 0.1=[$\mathrm{id}_1+\mathrm{const}$]	0.9	0.1[$\mathrm{id}_1+\mathrm{const}$] + 0.125[$a+\mathrm{const}$]
$c+10$	0.25=[$c+10$], 0.2=[id_1+10], 0.175=[c, +, 10], 0.125=[$c+\mathrm{const}$], 0.1=[$\mathrm{id}_1+\mathrm{const}$]	0.4	0.1[$\mathrm{id}_1+\mathrm{const}$]
$\frac{1}{a+3}$	0.16667=[$\frac{1}{a+3}$], 0.13334=[$\frac{1}{\mathrm{id}_1+3}$], 0.08333=[1, $a+3$], 0.06666=[id_1+3], 0.08334=[$\frac{\mathrm{const}}{a+\mathrm{const}}$], 0.04167=[+, 3, a, $a+\mathrm{const}$], 0.06667=[$\frac{\mathrm{const}}{\mathrm{id}_1+\mathrm{const}}$], 0.05833=[$a+\mathrm{const}$], 0.04667=[$\mathrm{id}_1+\mathrm{const}$]	0.89996	0.03333[$\mathrm{id}_1+\mathrm{const}$] + 0.08333[$a+3$] + 0.06666[$\mathrm{id}_1+3$] + 0.04167=[$a+\mathrm{const}$]
$\frac{1}{b+3}$	0.16667=[$\frac{1}{b+3}$], 0.13334=[$\frac{1}{\mathrm{id}_1+3}$], 0.08333=[1, $b+3$], 0.06666=[id_1+3], 0.08334=[$\frac{\mathrm{const}}{b+\mathrm{const}}$], 0.04167=[+, 3, b, $b+\mathrm{const}$], 0.06667=[$\frac{\mathrm{const}}{\mathrm{id}_1+\mathrm{const}}$], 0.05833=[$b+\mathrm{const}$], 0.04667=[$\mathrm{id}_1+\mathrm{const}$]	0.39996	0.03333[$\mathrm{id}_1+\mathrm{const}$] + 0.06666[$\mathrm{id}_1+3$]
$\frac{1}{a+5}$	0.16667=[$\frac{1}{a+5}$], 0.13334=[$\frac{1}{\mathrm{id}_1+5}$], 0.08333=[1, $a+5$], 0.06666=[id_1+5], 0.08334=[$\frac{\mathrm{const}}{a+\mathrm{const}}$], 0.04167=[+, 5, a, $a+\mathrm{const}$], 0.06667=[$\frac{\mathrm{const}}{\mathrm{id}_1+\mathrm{const}}$], 0.05833=[$a+\mathrm{const}$], 0.04667=[$\mathrm{id}_1+\mathrm{const}$]	0.3	0.03333[$\mathrm{id}_1+\mathrm{const}$] + 0.04167[$a+\mathrm{const}$]
$\frac{1}{c+10}$	0.16667=[$\frac{1}{c+10}$], 0.13334=[$\frac{1}{\mathrm{id}_1+10}$], 0.08333=[1, $c+10$], 0.06666=[id_1+10], 0.08334=[$\frac{\mathrm{const}}{c+\mathrm{const}}$], 0.04167=[+, 10, c, $c+\mathrm{const}$], 0.06667=[$\frac{\mathrm{const}}{\mathrm{id}_1+\mathrm{const}}$], 0.05833=[$c+\mathrm{const}$], 0.04667=[$\mathrm{id}_1+\mathrm{const}$]	0.19	0.03333[$\mathrm{id}_1+\mathrm{const}$]

because it is applied to all of them. Another innovation the handling function for the weight of the attributes (Section 4.2.5). It takes an opposite approach to the other unification methods and increases the weight of the expressions that retain their important attributes.

To summarize, there are 4 coefficients in MIaS which affect the resulting weight of indexed expressions:

- level coefficient $l = 0.5$,
- variable unification coefficient $v = 0.6$,
- number unification coefficient $c = 0.8$,
- attribute coefficient $a = 1.2$.

There is a number of possible combinations in how we want to compute the distance of the generalized formulae from the original formula and have the results ordered accordingly. And there are many criteria to be considered when determining the values of the factors. In formula words, an expression composed of n nodes found in a certain *level* within an original formula is indexed with a final weight

$$w = \frac{l^{level}(1 + Cv + Vc + vc + Aa)}{n}$$

where C, V and A are values 0 or 1 depending on whether a formula has its constants unified ($C = 0$), variables unified ($V = 0$) and whether it contains important attributes ($A = 1$). See Table 4.1 on the preceding page for details.

4.2.7 M-terms

The final stage in the preprocessing of the mathematics contained in documents is producing character strings representing formulae so they can be stored by the indexing core. MIaS transforms MathML XML nodes to linear strings using brackets and prefix notation. For example a formula $\frac{1}{x^2}$ with its Presentation MathML representation

```
<math>
  <mfrac>
    <mn>1</mn>
    <msup>
      <mi mathvariant="bold">x</mi>
      <mn>2</mn>
    </msup>
  </mfrac>
</math>
```

is rewritten to `mfrac(mn(1)msup(mi[mathvariant=bold](x)mn(2))).`

Every formula in MIaS is coupled with a weight that was computed during the preprocessing. Ordered pairs of string representations of formulae and their weights – (string formula, weight) – we call **M-terms**. They might be usable not just for indexing and searching but also for other applications. In fact they are just another representation of formulae in documents and can even be encoded together with regular MathML within an appropriate tag.

Scalability experiments showed that the size of the index can be considerably lowered by substituting MathML element names, attribute names and common attribute values, which are constantly repeating in expressions, with single characters based on a dictionary. MIaS uses such a dictionary to compact the string representations of formulae. The resulting form of the previous example after applying the dictionary is `F(N(1)J(I[V=B](1)N(2))).`

4.3 Searching

When a user writes and posts a query to the system it is important it passes the same processing as in the indexing phase for the system to be able to match indexed tokens, whether it is stemming regular text or transforming query formulae to M-terms. There is however one exception in the processing of math – query formulae are not tokenized. Tokenization gives the system the ability to find a match inside an expression, formulae are therefore tokenized before they are indexed. This behavior is unwanted when searching. A user is most probably not expecting to find subparts of his queried formula but conversely, wants his expression to be found inside different formulae.

Several (4 or 5) M-term representations of each query formula are produced during preprocessing. Generated weights are not used and only formulae strings connected to the textual part of the query are passed to the searching core in the following manner: (formula$_1$ $\vee \ldots \vee$ formula$_n$) $\wedge$ (term$_1$ $\vee \ldots \vee$ term$_n$). Users can override this default query composition by manually stating preferences for query tokens to occur in the matched documents by using AND and OR operators.

In Section 4.2.6 a weighting scheme for indexed formulae was described. The final scores of matched and retrieved formulae are completely finalized when searching. MIaS additionally sets the boost for query formulae according to their complexity. This has two reasons: one is to counterbalance text and formula parts of the query since matched expressions can be indexed with relatively low weight, and secondly to balance weights between different query formulae. The bigger a formula in the query, the bigger weight it should have in the resulting score of matched documents – the boost factor is the number of the query expression's nodes.

The final scores of matched documents are computed based on several factors. When searching for mathematical formulae, their precomputed indexed weights need to be considered in the score of the document. With MIaS being built on the top of Lucene indexing engine, it makes use of that scoring function[3] (described in detail at [6]) and adds another parameter to it – weight w_t of the term t if t is a formula. The final scoring function which computes the score of a hit document d by query q with each query term t is as follows:

$$score(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \\ \cdot \sum_{t\ in\ q} \left(tf(t \text{ in } d) \cdot avg(w_t) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d) \right)$$

If a document contains the same formula more than once (each occurrence can have a different weight assigned), the average value of all the weights is taken into consideration, therefore $avg(w)$.

4.4 MathML Processing

MIaS aims to process real world documents. It was primarily focused on Presentation MathML markup, which can be relatively easily converted from the format most used by authors, namely TEX and its variations. Its biggest advantage, availability, is negated by several downsides. For machine processing it contains a lot of unnecessary markup which is used mainly for nice rendering. There is very little semantics in the encoded formulae, i.e. element `<mi>f</mi>` can not be told whether it is a variable or a function name or the name of some constant. One can hardly tell arguments of an operator or a function.

There are many initiatives focusing on putting more semantics into electronic documents. Several different markups have been developed for this purpose (S-TEX, OpenMath for mathematics, Content MathML, etc.). With this effort, the quality of tools which are able to work with semantics is improving. Documents are being enriched with metadata and semantics in general. Such tools are also converters of mathematical notation, for example LATEXML which is able to produce mixed Presentation and Content MathML from LATEX sources. With the help of this tool, a corpus of semantically enriched scientific documents is being built by converting the arXiv corpus [25].

To extend precision coming from the semantic character of Content MathML, MIaS expanded its capabilities and is now able to index Content MathML in the same way as its presentational part. Versatility in the XML processing in the math preprocessing module showed no exhaustive effort was needed to achieve

3. Class Similarity. Apache Lucene. 4 Jan. 2013 `http://lucene.apache.org/core/3_6_1/api/core/org/apache/lucene/search/Similarity.html`

this. Preliminary tests indicate indexing Content MathML has a canonicalization effect, since there is a smaller possibility to generate different structures due to the lack of semantically unnecessary elements. M-terms produced from Content MathML trees come into effect exactly when Presentation MathML M-terms do not match due to some slight differences.

From the implementational point of view, presentation-sourced M-terms are stored in the different document's index field than content-sourced terms fields. Document mathematics is processed in the two-pass fashion; in each one, a different MathML part is processed. The same weighting factors are used for both encodings, which remains subject to further evaluation.

When querying the system it tries to retrieve both types of MathML from the query by subsequently passing it to MathTokenizer with different MathML type parameter (similar to the indexing phase) and depending on the result, it searches corresponding document fields for matches. To be able to have queries in Content MathML, there needs to be a powerful converter on the user side, such as LaTeXML. Previously used Tralics is incompatible in such a setup. Of course, users can still copy and paste Content MathML from a different source when building their query.

4.5 Other Features

One of the important properties of an IR system which aims to be used in real world is a high level of user comfort. A system can perform excellently in retrieving relevant documents, but when a user has problem accessing and consuming such a system's capabilities, the system can be perceived negatively. There are many ways and features how to accomplish a good level of usability as we can see on todays most used search services.

4.5.1 Generating Snippets

One such a feature is match highlighting and displaying a snippet of a document where important matches to query terms occurred. It gives the user a quick glimpse on what parts of his query were found in a document, and it is also an instrument for early evaluation of document's relevance. Without match fragments a system can return only a document title for each hit, possibly some metadata and a few of the first sentences as an excerpt of its content. These data may not be representative and may not justify why this document is a hit. It is left to the user to evaluate relevance of the results by going through them.

There are several possible strategies for snippet generation. The first is to store the original terms in the index – they can be retrieved quickly in search time and displayed to the user but only in the form of separate words that matched portions of the query. Another downside of this approach is the

space requirements for the index. A different approach is to store document fragments in a separate index, match document is re-queried at search time in the fragments index to retrieve snippets of significant matches. The upside of this approach is that there are not any additional I/O operations to retrieve snippets, but the shortcoming is a very large secondary index, usually larger than the primary one [29]. The third approach, as implemented by MIaS, is to index terms with their positions in the original document. When retrieving results, the system needs to open the source document, lookup matched term by its position and generate a snippet. This slows down hit list generation, but requires no additional storage.

The MIaS system is capable of showing match snippets to the user. As opposed to regular text search, math preprocessing changes the visual of expressions quite heavily (consider Section 4.2.7), so it can not implement the first of the solutions described above. MIaS opted for the third solution and since it weighs every formula, when a query hits a document the most significant terms can be easily retrieved together with their positions. The position is a sequence number of a hit formula among other document formulae; other document terms are not considered. Derived and extracted M-terms hold the same position information as their original formula, therefore when a query matches a subformula, the system can only highlight the whole original formula.

When retrieving search result terms which caused a document to match, they are ordered according to their weight. Two most significant terms are located in a document by their position and they are added to the final snippet with surrounding fragments. These surroundings extend to the start or the end of the sentence, or to the next or previous XML structure within a sentence. This is for the resulting snippet not to be very large as this can be disturbing to look at the results page. A possible improvement to this solution would be to index a unique identifier of each expression, for example an *id* attribute, for the search-time parser to be able to search for the right formula more quickly. MIaS however does not rely on every document source to have such identifiers, therefore it uses positions of formulae terms.

4.5.2 Optimisations

Initial scalability tests [23] showed there is room for the system to scale better with an increase in the number of documents. As a result of many little optimizations, MIaS is now able to use the computing potential of today's machines by using parallelization. The system uses threads for preparing and preprocessing documents, especially for the most time consuming preprocessing of math formulae. As writing to the index is a write I/O operation using only one resource, it is not able to be done in more than one thread. The results of preprocessing from individual threads are returned to the main thread which creates the index. The system therefore needs to spawn a preset number of

threads and poll them in a cycle to see whether they have finished. This creates an overhead but still makes an advantage to a single threaded application.

Another effort was to optimize the size of the index. Since MIaS indexes several generated representations for each math expression it creates demands to the index storage capacity that are out of the ordinary for regular text indexes. One step is to compact the final form of M-terms (see 4.2.7). The second step is to use a number data type with the smallest possible range for storing weights. This may seem an insignificant amount, but using 2 byte short number instead of 4 byte float number, which MIaS uses for weight, creates a 4 GB difference when 2 billion formulae are indexed, as is contained in about 300k arXMLiv documents.

A similar optimization was also performed to reduce the storage requirement of input documents. Several thousands of files can occupy a large storage space. MIaS was adapted to this and is able to process documents in a compressed state. `FileExtDocumentHandler` is able to detect if the received file is a zip file, open it and get input streams of files contained within and handle them as normal files according to their extension. So far, there was no need to handle different file types more strongly.

4.6 Web Interface

The MIaS Project is meant to be usable with real world documents and also focuses on being easy to use for real world users. For this purpose a web interface called WebMIaS has been developed. Technically it is a separate Java project which includes MIaS and uses it for searching previously created indexes.

It uses a simple and straightforward interface for writing queries – only one input text box, where users can write text and math queries at the same time. Very first version of WebMIaS [9] only accepted math in MathML notation, later modifications allowed posting TEX which were converted on-the-fly by Tralics. A user had to specify the notation by selecting one from the select menu. The latest version is able to auto-detect TEX math that is surrounded in $ signs, and converts it on-the-fly by more powerful LATEXML. For even better usability, TEX queries are converted on-the-fly to presentation MathML by the SnuggleTeX[4] converter and rendered for visual verification by MathJax[5].

Another new feature is that one instance of WebMIaS is able to search in multiple indexes by selecting an index from the menu. This is useful if we do not want to mix different document bases into one index. It was used extensively during the development by comparing results for the same queries in different

4. SnuggleTeX. School of Physics & Astronomy, The University of Edinburgh. 4 Jan. 2013 `http://www2.ph.ed.ac.uk/snuggletex/documentation/overview-and-features.html`

5. MathJax. 4 Jan. 2013 `http://www.mathjax.org/`

indexes, but it is believed to be useful for regular use as well. A downside of this feature is that there is only one version of MIaS behind the interface so it can not be used to test different indexes created by different versions of the programs used, e.g. MIaS or MathMLCanonicalizer.

The appearance of the result list is essentially unchanged from its first version. One result entry consists of a document title, which in case of arXMLiv documents is a hyperlink to its original arXiv resource. There is also a hyperlink pointing to the local document that was used for indexing. Then there is a final score of matched documents for comparison mainly for development purposes. The main addition to the appearance of result entries and to the system's usability is a snippet highlighting the most significant matches in the document (see Section 4.5.1). Formulae in the snippets are rendered using MathJax[3]. Furthermore debugging information has been added to the interface showing the Lucene form of the query and a detailed explanation of every hit document. It is activated by checking the "debug" checkbox.

Search services of MIaS can also be accessed remotely since WebMIaS among its web interface also supports a RESTful web service for searching. One can basically use the searching as he would using the web interface by providing several parameters to the service: a string containing the query, index number to search in, starting offset and a limit for the number of the results to be returned. The service returns searching time, the total number of hits and a set of hits each consisting of title, id, additional info and a match snippet.

The additional web service that used MIaS which was initially a separate project was incorporated into WebMIaS in order to have a more integrated module instead of several little ones. This web service was called MIaS4Gensim and it allowed remote callers to transform MathML formulae to MIaS weighted M-terms and vice-versa. It was mainly used for experiments with Gensim similarity computational software [17] to include mathematics into its computations, however we believe that this functionality may also be useful for other applications. The interface is very simple: for given MathML formula the service returns a list of pairs consisting of M-terms string and a weight floating point number.

To improve the accessibility of WebMIaS, OpenSearch [2] standard is supported. OpenSearch is a set of formats and rules which a search engine is described by in order to be easily queried by, possibly automatic, clients. Search results are then easily shareable across the web. To be OpenSearch compatible a system needs to publish OpenSearch description XML document in which several information are stated: name of the search engine, contact to the developer, tags describing the system, language, input and output encoding and most importantly, examples of queries and the parameters which can be posted by clients. Most common parameters are search terms, start index number (index of a first result a client wants to retrieve) and limit (the number of documents to return). The document is extensible with custom elements which have to be

prefixed with a declared namespace prefix and the namespace url must point to a custom descriptive document.

4.7 Refactoring and Packaging

The system has been considerably refactored to make the scattered source code created by ad-hoc development of new features, more consistent and readable. This affected mainly the searching part of the system where a lot of the important functionality was initially added to the WebMIaS client. Now it has been moved to the MIaS core, which exposes more readable and easy to use API. Also since canonicalization is now an integral part of math preprocessing, several methods in MathTokenizer which tried to prepare and canonicalize the formulae by themselves are left out as well as some of the core methods were simplified.

The whole creation of Lucene documents in MIaS has been revised as well. This has been done mainly to support zip files and for different types of inputs to be easily addable in the future. `FileExtDocumentHandler` is a strategy to create Lucene documents which are indexed. There is a `DocumentSource` inteface, which represents the source of the input data. It can be a file (`FileDocument`) or a zip file entry (`ZipEntryDocument`). Different sources have different methods for accessing their input streams. Implementations of `MIaSDocument` need to access these methods in order to extract all the needed information from the source. Additionally, every `DocumentSource` implements a creation of a default Lucene document.

MIaS and WebMIaS applications are packaged as standalone *jar* and *war* Java applications. Their build, dependencies as well as packaging is managed using Maven system[6]. MIaS still accepts commands for indexing and searching from a command line. WebMIaS needs a web container for running. These two applications are referential implementations of a math-aware Lucene-based search system and are not expected to be deployed as a standalone application on the production level, however, it is possible. The first version of the system was created in the course of my Bachelor thesis. The second version will be the first publicly distributed version. This work has moved the application very close to this milestone.

It is more likely that only math processing capabilities of the system will be used in some productions which use Lucene or Solr, similarly to EuDML. This module (see Section 4.2), called MIaSMath, was therefore packaged separately and is distributed together with the API documentation for Lucene implementation as well as with Solr integration guidelines.

---

6. Apache Maven Project. The Apache Software Foundation. 4 Jan. 2013 `http://maven.apache.org/`

5 Evaluation

In Chapter 2 we looked at the history of IR evaluation. Methods and measures that has been established throughout the years were described in Section 2.3. None of them, however, have been applied to mathematics IR so far. The first ever collection that aimed to evaluate math IR systems' efficiency was MREC [10] and the results of the MIaS system have been reported [23]. The collection that is also applicable for evaluating the effectiveness was created for the MIR Happening, but none of the measures have been reported so far.

We outlined different use cases and scenarios in math retrieval and linked them to the techniques for creating test collections as well as the measures that could be observed. We learned that to create a useful collection, mathematician's knowledge should be participating to be introduced into the collection or to the evaluation process.

Considered the things mentioned above any ambitions to create an own collection for the evaluation of MIaS were dropped. With the increasing interest in MIR and its evaluation (Sections 2.2.4, 2.2.5) it also feels redundant to try to create the test suite at own effort. This work leverages the collection created for the MIR happening. Despite it not being very large it can show basic properties of a system underlined with some of the basic measures, more so if they have not been computed at the happening.

The MIR-sandbox collection, which is used for evaluation, consists of 10,000 non-trivial math documents extracted from arXMLiv corpus. The set of queries devised by three mathematicians is divided into three categories as described in Section 2.2.4 with one difference – formula search queries are also searched in the sandbox collection, since provided matching documents are found in this collection as well. There are two groups of these three types of queries. Every query from the first group has a matching document in the sandbox collection. The second group consists of open-ended queries for problems that the mathematicians found interesting, but for which no matching document is known to be found in the collection. These were not carried at the MIR 2012 happening.

Even though the system was already queried with the queries from the collection at the happening, a lot of work has been done since then. Evaluation in this work captures the state of the system at the end of writing this thesis. Detailed analysis is shown to explain why each of the queries did or did not match a designated target document. Open information retrieval query is not tried as it is an extremely hard problem (see Appendix C) and needs to be tested with the cooperation of the query's author.

5.1 Queries

Queries are shown in source TEX notation as well as in their rendered forms. The question marks in front of some formula elements denote query variables. It means that the element may not be present in the document in that exact form. Queries in MIaS do not need to specify explicitly which variables are query variables. MIaS searches for their exact as well as unified form implicitly. The name of the correct result document file is displayed under each query.

5.1.1 Formula Search

1. Recollect a historical formula, such as:

TEX: `\sqrt{2} = 1 + \frac{1}{3} + \frac{1}{3\dot 4} ? - \frac{1}{3\dot 4\dot 34}`
Math: $\sqrt{2} = 1 + \frac{1}{3} + \frac{1}{34} - \frac{1}{3\dot 4\dot 34}$
Result: f005795.xhtml

The above query returns no results. After a look into the correct result document, we see that the formula that should match our query is written differently. It uses regular dots as times operator and therefore 3.4 and 3.4.34 were converted as a single number "three point four" and a senseless "three point four point thirty-four". There is no smaller piece of the expression specific enough to retrieve the desired document.

After querying the original TEX form of the formula `\sqrt{2}=1+\frac{1}{3}+\frac{1}{3.4}-\frac{1}{3.4.34}`, the correct result is returned.

2. Retrieve instances matching:

TEX: `B_{p+n} = B_n + B_{n+1} \bmod p \ \text{for all}\ n=0,1,2,\dots`
Math: $B_{p+n} = B_n + B_{n+1} \pmod p$ for all $n = 0, 1, 2, \dots$
Result: f005794.xhtml

Presuming the visual form of the query is the form we want to find, there is already a problem in the query. It uses a macro `\bmod` to typeset `mod` in roman. We need to substitute it with `\mathrm`.

Even after the substitution, no results are found. We need to look in the correct document. There are several problems causing the differences between the query and the formula in the document. There is an $\equiv$ sign in the document instead of equal sign in the query. The whole formula from the document is strangely structured; elements from the beginning until the $n = 0$ part are encapsulated in one `mrow` leaving out other elements that are on the same logical level. And there is also a sentence ending dot contained as the last element of the formula.

In the content part of the query there is wrongly an `<and/>` element placed in front of the `<equivalent/>` element. Also, there is a similar problem with the formula being split through $n = 0, 1, 2, \ldots$ into two separate structures. Finally, the `...` is converted to `<ci>normal-...</ci>`.

The differences mentioned above cause searching for the whole formula impossible. Querying only for the part `B_{p+n}` returns the correct result at position one.

3. Find examples of the use of the below metric:

TEX: `S(g)= \frac{s(g)-s_{\text{min}}}{s_{\text{max}}-s_{\text{min}}}`
Math: $S(g) = \frac{s(g)-s_{\text{min}}}{s_{\text{max}}-s_{\text{min}}}$
Result: f005796.xhtml

The above query return no results. The reason is the incorrect use of the `\text` command to output roman letters in the formula. The converter generates `<mtext></mtext>` element which is wrong as "max" and "min" are identifiers in this formula. Changing `\text` for `\rm` creates the correct query and finds the desired result at position one as the only result.

4. Find Cardy's formula:

TEX: `\frac{3\Gamma(2/3)}{\Gamma(1/3)}\eta^{1/3}\,\rule{0pt}{10pt}_2 F_1(1/3,2/3,4/3;\eta)`
Math: $\frac{3\Gamma(2/3)}{\Gamma(1/3)}\eta^{1/3}\,{}_2F_1(1/3,2/3,4/3;\eta)$
Result: f005692.xhtml

The given query does not find any result. Even removing suspicious part `\rule{0pt}{10pt}` or trying to search for the subformulae $\frac{3\Gamma(2/3)}{\Gamma(1/3)}$ or $3\Gamma(2/3)$ does not retrieve the satisfactory result. A look into the desired document shows that not even remotely similar formula is contained in it. This is most probably a mistake in the test collection.

5. Retrieve instances matching:

TEX: `a?x^2+b?y^2`
Math: $ax^2 + by^2$
Result: f004977.xhtml

The query returns two results none of which is the correct document. The actual variables in the document are x_1 and x_2 and additionally, the query is only a subpart of the original expression $p_1 = ax_1^2 + bx_2^2 + \varepsilon_1$. The formula is not found for two reasons. MIaS can not unify a variable with an index to a simple variable; $x + y$ is unified to $id1 + id2$ but $x_1 + x_2$ is unified to $id1_1 + id1_2$

which is different from the expression structure point of view. The other reason is that the query is not a logical subpart of the original formula – searching only for two of the three addends is impossible in MIaS.

6. Retrieve instances matching:

TEX: `\frac{e^2+3}42^{?l\choose 2}?n^?l`
Math: $\frac{e^2+3}{4}2^{\binom{l}{2}}n^l$
Result: f004150.xhtml

The query finds the correct document as the only result. The Presentation MathML part of the query caused the match with the document. There are differences in the Content MathML: `l\choose 2` in the query is converted to

```
<apply>
    <csymbol>binomial</csymbol>
    <ci>l</ci>
    <cn>2</cn>
</apply>
```

while in the document there is an empty element `<ci/>` instead of the `<csymbol>binomial</csymbol>`.

7. Retrieve instances matching:

TEX: `?P\in \sum_{i=1}^r \Z ?{P_i}`
Math: $P \in \sum_{i=1}^{r} \mathbb{Z} P_i$
Result: f004102.xhtml

The query uses a macro to typeset the number set Z. After the extraction of the macro, the query becomes `P\in \sum_{i=1}^r \mathbb{Z} {P_i}`, but still does not found the document. Again, there are several differences preventing the match. The query converter encloses the $\mathbb{Z}P_i$ part into additional `<mrow></mrow>` element, which seems correct as it is a single parameter of the sum function. The Content MathML part of the query does not help the situation. There is a `<in/>` element as opposed to `<ci>∈</ci>` element in the document, which are both valid forms. The other difference is that there is correctly a $\mathbb{Z}$ in the query, but only simple Z in the document Content MathML.

Searching for the largest single structural unit which does not cause any troubles, $\sum_{i=1}^{r}$, causes the system to return 1,045 results, first of which is the correct document.

5.1.2 Full-text Search

Handle the following textual queries:

1. Where can I find the formula for free cumulants in terms of the symmetric group?

Result: f005793.xhtml
Query: "free cumulants" AND "symmetric group"
The only result is the correct document.

2. Aren't there some newer special polynomials involved?

Result: f005793.xhtml
The question here is the intention of the query author: whether the query is only a supplementary question to the first one or a completely new one asking about any newer special polynomials.
Query: "free cumulants" AND "symmetric group" special polynomials
If we only add the keywords "special" and "polynomials" to the first query to verify whether the document talks about any special polynomials, the only correct document is still returned with the verification of the keywords being highlighted.
Query: special polynomials
The query returns 2,373 results; the correct document at position one.
Query: "special polynomials"
Returns only one result f000391.xhtml but it must be verified with a mathematician to determine if it is relevant to the information need.

3. Also, Kerov polynomials and zonal polynomials

Result: f005793.xhtml
Query: "Kerov polynomials"
The only result is the correct document.

The judges copy of the MIR Happening instructions document wrongly states f005795.xhtml as the correct result for the first two full-text queries. A look into the document shows it does not mention any cumulants or symmetric groups.

5.2 Effectiveness

We will consider a search successful even if the query was not exactly the same as in the specification document, partly because not all of the notational details were correct and partly because we think the query does not find a complete

match in order to retrieve the desired document. We will omit the incorrect fourth formula search query in the effectiveness calculations.

We can compute basic precision and recall values with the presumption there was always only one relevant document per query as we do not have the resources needed to judge the relevance of other documents in cases when more than one document were retrieved.

Table 5.1: Results of the queries with their precision and recall

Query	Results retrieved	Relevant docs retrieved	Precision	Recall
Formula 1	0	0	0	0
Formula 2	207	1	0.0048	1
Formula 3	1	1	1	1
Formula 5	0	0	0	0
Formula 6	1	1	1	1
Formula 7	1,045	1	0.00096	1
Full-text 1	1	1	1	1
Full-text 2	1	1	1	1
Full-text 3	1	1	1	1

Based on these measurements, the average precision of the system is 0.56 and the average recall is 0.78. The combined balanced F-measure of the system therefore is

$$F = 2 \times \frac{0.56 \times 0.78}{0.56 + 0.78} = 0.65$$

Since the test collection is the type of collection with injected correct documents, the reciprocal rank seems to be a more appropriate measure.

Table 5.2: Results with the rank and the reciprocal rank of the correct result

Query	Correct result rank	Reciprocal rank
Formula 1	0	0
Formula 2	1	1
Formula 3	1	1
Formula 5	0	0
Formula 6	1	1
Formula 7	1	1
Full-text 1	1	1
Full-text 2	1	1
Full-text 3	1	1

If the MIaS retrieved the correct result, it had the rank 1, which promises a good overall score. The mean reciprocal rank for the system then is:

$$MRR = \frac{0+1+1+0+1+1+1+1+1}{9} = 0.78$$

The character of the collection and the character of the result sets for the queries are unsuitable for computing any other measures. For example precision at p requires more results to be retrieved and their relevance judged.

5.3 Efficiency

Efficiency of the MIaS system with the above stated effectiveness level was measured using the NTCIR corpus consisting of 100,000 documents. The measurement was made on a on 448 GiB RAM, eight 8-core 64bit processors Intel XeonTMX7560 2.26 machine using 64 threads for the preprocessing of the indexed documents. The measured data are displayed in Table 5.3.

Table 5.3: Measured data for the efficiency evaluation

	Indexing times [min]		Formulae		Index	Av. query time [ms]	
Docs.	Wall clock	Total CPU	Input	Indexed	size [GB]	Core	Total
10,000	28.8	159.7	7,327,283	155,192,904	3,1	188.2	495.4
20,000	58	325.2	14,736,285	311,258,718			
30,000	85.1	474.2	21,877,907	463,281,808			
40,000	111.5	616.1	29,299,122	618,586,152			
50,000	146.1	821.8	36,801,976	779,487,671	15	182.5	484.1
60,000	177.1	999.4	44,179,606	938,538,811			
70,000	203.1	1,143.6	51,394,938	1,088,869,124			
80,000	231.5	1,306.6	58,633,240	1,241,466,398			
90,000	261.2	1,475.4	66,065,698	1,398,541,881			
100,000	291.8	1,649.0	73,428,180	1,556,839,999	30	199.1	601.9

Figures 5.1, 5.2 and 5.3 show the measurements and their reliance on the number of the input documents.

In order to compute average query times, the times of MIR Happening queries were measured on 10k, 50k and 100k document indexes. The query set consisted of 7 math and 4 text queries, some of which did and some of which did not find any results. Core search time is the time needed for the full-text core to return the hit documents. Total query time is the core search time plus any additional time needed to display the results, i.e. retrieving details about the hit and extracting snippets. In the current setup with TEX to MathML converter converting queries on the client side through a web service, approximately 1 extra second is added to every query. This is not shown in the above table and the diagrams as it is a constant which is subject to further optimization.

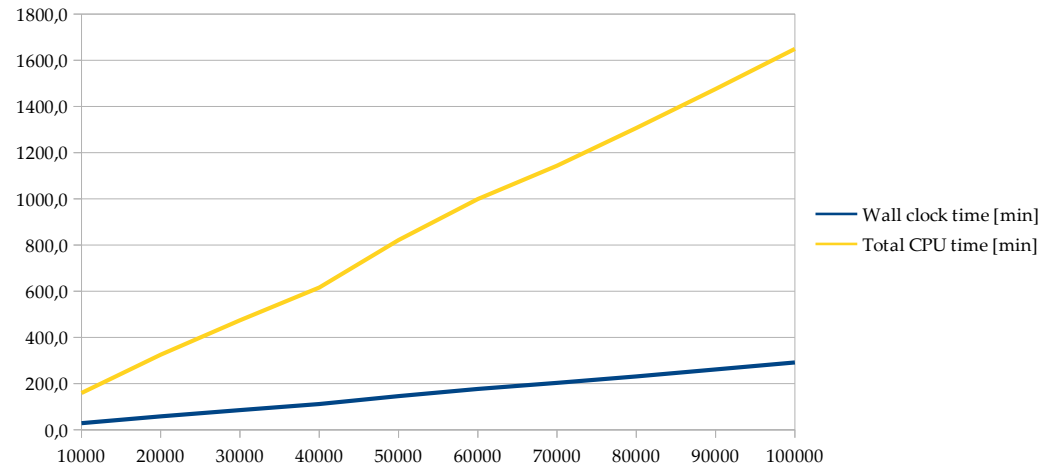


Figure 5.1: Indexing times

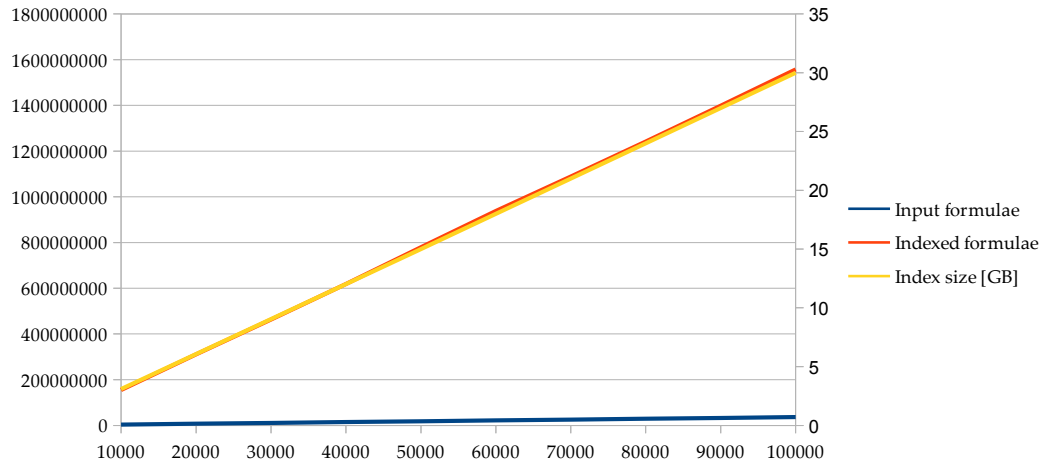


Figure 5.2: Input and indexed formulae and index size (right Y axis)

5.4 Conclusion

In this Chapter the effectiveness and efficiency of the MIaS system was evaluated. We discovered, that despite the canonicalization the system is still quite sensitive to the formulation of queries. This is because the canonicalizer so far normalizes only different variations of the correct notations, but does not fix errors and inappropriate uses of TEX or MathML elements on the input. For this we have not yet devised a solution.

After correcting the queries, the system performed, we think, quite well. In 7 of the 9 queries it retrieved the desired document at the first position; most of the time it was the only result as well. This led to a mean reciprocal rank of

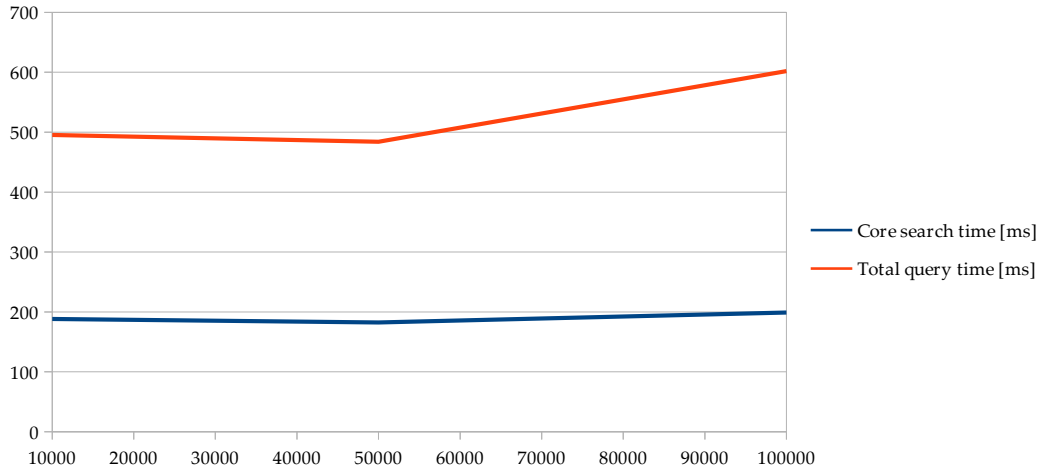


Figure 5.3: Average query times

0.78 and an average combined balanced F-measure 0.65. The best way to know whether these numbers are good enough or not would be to compare them with another system. This will be done at the NTCIR math task.

Our efficiency evaluation showed that the system's properties are reasonable in terms of both indexing and query times. When using multiple threads, the total CPU time needed to index a document collection can be at least halved; in our case the wall clock time was five times shorter than the total CPU time. The diagrams show that the system scales linearly in terms of indexing time and index storage requirements with respect to the rising number of documents and the number of the formulae contained in them respectively. The factor of indexed documents to the number of input documents is averaged at about 42, which means there are 42 formulae generated and indexed for one input formula. This factor has doubled since the last measurements [23] which is caused by indexing both Content and Presentation MathML. The same applies to the index size. The index size is above average compared to the regular text indexing, which is caused mainly by the indexed-input formulae factor.

The query time, however, has quite a stable character. As can be seen in the diagram 5.3, the average core query time increased only by about 10 ms from searching 10k documents to searching 100k documents. This is only a 5% increase in the query time when there is a 1000% increase in the number of documents searched. What slows the queries down is the conversion of TEX parts of the queries and retrieval of the match snippets. This can be optimized by changing the strategy of retrieving resources from the outside of the system which will result in better user comfort and will be subject to further evaluation.

6 Conclusion

Information retrieval is a key technology in accessing the vast amount of data there is on todays World Wide Web. It is also one of the key technologies in knowledge management. Developments in information retrieval began already in the pre-computer era and still continue today. The evaluation of IR has been an integral part of IR development. It is the moving force which influences the advances in the field. Several annual conferences that deal exclusively with information retrieval testify to the dynamism of the field.

Mathematics retrieval is a new type of information retrieval. It focuses on searching structured mathematical data to simplify the knowledge management in specialized portals that provide this type of information. The evaluation of MIR has not been dealt with until very recently. Raising interest in MIR and its evaluation has so far resulted in two organized events in the fashion of already established evaluation practices in other types of IR.

This work overviews the history of IR and its evaluation as well as the techniques that have been developed. Effectiveness evaluation is based on the notion of relevant and non-relevant documents with respect to a query. The key to evaluate an IR system is to devise an evaluation collection composed of a document collection, a collection of queries and the relevance judgments for the pairs of documents and queries. This basic scheme can by applied to any type of retrieval including mathematics. The main difference is the preparation of the test collection since evaluating the relevance of mathematics can be done only by mathematicians.

The development of Math Indexer and Searcher, a MIR system, is summarized in this thesis. It was created as a part of my bachelor thesis and has been evolving since then. It was already integrated into real production system – the EuDML search[1], and further deployments are expected in the future as the system will go public open-sourced. One of the goals is to have MIaSMath including MathTokenizer included in the official Lucene Contrib distribution[2]. A lot of work has been done in the course of writing this thesis, mainly to organize and refactor the code that has become shattered during two years of ad-hoc development. Also a lot of work has been done in order for the system to perform as well as possible for the MIR Happening as well as the NTCIR math task.

This can be observed in Chapter 5. MIaS has been evaluated using the MIR Happening test collection. A detailed analysis of the effectiveness as well as the efficiency evaluation is provided. The results of the NTCIR math task will

1. Advanced Search. EuDML. 4 Jan. 2013 `http://eudml.org/search.action`
2. Lucene Contrib. The Apache Software Foundation. 4 Jan. 2013 `http://lucene.apache.org/core/3_6_2/lucene-contrib/index.html`

provide comparable results across several different systems and should confirm the good performance of MIaS.

MIaS is a referential implementation of a mathematical search build on the top of the full-text indexing library Lucene. In this respect, it could also improve the usability mainly for system administrators, as the indexing phase of the system is still operated from a command line. Making the whole application web-based with easily understandable interface for index administration and system configuration would be a welcome improvement. But the main goal is to make the math processing part of the system as robust, effective and efficient as possible. There is still a lot of work to be done in order to achieve this. Our focus is now primarily on the canonicalizer which should make the system less vulnerable to different types of inputs.

Other future work includes exploiting semantic information for searching, using text as well as extracting text descriptions of formulae in order to disambiguate them [22]. Making personalized searches is very popular these days. MIaS could personalize the searching according to user's previous attempts as well as the similarity of the results that have been further explored. There is already a functional technology for computing document similarity including mathematics that could be integrated into MIaS [17]. Imminent work includes the evaluation of the NTCIR math task results which is expected to result in more detailed tasks to increase the system's F-measure. A conversion to the newest version of Lucene, 4.0, which promises improved search performance as well as index compression will be another step. Relevant information about the system, latest news, links to working demos as well as relevant publications are located on the project webpage `https://mir.fi.muni.cz/mias`.

With many future plans to make the system as good as possible, with an already existing integration and with more to come, Math Indexer and Searcher aims to become the first widely used mathematical search system.

Bibliography

[1] Dominique Archambault and Victor Moço. Canonical MathML to Simplify Conversion of MathML to Braille Mathematical Notations. In Klaus Miesenberger, Joachim Klaus, Wolfgang Zagler, and Arthur Karshmer, editors, *Computers Helping People with Special Needs*, volume 4061 of *Lecture Notes in Computer Science*, pages 1191–1198. Springer Berlin / Heidelberg, 2006. http://dx.doi.org/10.1007/11788713_172.

[2] DeWitt Clinton, Joel Tesler, Michael Fagan, Joe Gregorio, Aaron Sauve, and James Snell. OpenSearch. 4 Jan. 2013 http://www.opensearch.org/Home.

[3] World Wide Web Consortium. W3C math home. 4 Jan. 2013 http://www.w3.org/Math/.

[4] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley, 2009.

[5] David Formánek, Martin Líška, Michal Růžička, and Petr Sojka. Normalization of digital mathematics library content. In James Davenport, Johan Jeuring, Christoph Lange, and Paul Libbrecht, editors, *24th OpenMath Workshop, 7th Workshop on Mathematical User Interfaces (MathUI), and Intelligent Computer Mathematics Work in Progress*, number 921 in CEUR Workshop Proceedings, pages 91–103, Aachen, 2012.

[6] The Apache Software Foundation. Lucene Scoring. 4 Jan. 2013 http://lucene.apache.org/core/3_6_1/scoring.html.

[7] Michael Kohlhase, Ştefan Anca, Constantin Jucovschi, Alberto González Palomo, and Ioan A. Şucan. MathWebSearch 0.4, A Semantic Search Engine for Mathematics. 7th International Conference on Mathematical Knowledge Management, 2008.

[8] Michael Kohlhase, Bogdan A. Matican, and Corneliu-Claudiu Prodescu. Mathwebsearch 0.5: Scaling an open formula search engine. In Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *Intelligent Computer Mathematics*, volume 7362 of *Lecture Notes in Computer Science*, pages 342–357. Springer Berlin Heidelberg, 2012.

[9] Martin Líška. Vyhledávání v matematickém textu (in Slovak), Searching Mathematical Texts, 2010. Bachelor Thesis, Masaryk University, Brno, Faculty of Informatics (advisor: Petr Sojka), https://is.muni.cz/th/255768/fi_b/?lang=en.

[10] Martin Líška, Petr Sojka, Michal Růžička, and Petr Mravec. Web Interface and Collection for Mathematical Retrieval: WebMIaS and MREC. In Petr Sojka and Thierry Bouche, editors, *Towards a Digital Mathematics Library. Bertinoro, Italy, July 20–21st, 2011*, pages 77–84. Masaryk University, July 2011. http://hdl.handle.net/10338.dmlcz/702604.

[11] Thomas Mandl. Recent developments in the evaluation of information retrieval systems: Moving towards diversity and practical relevance, 2007.

[12] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[13] Bruce R. Miller and Abdou Youssef. Augmenting presentation mathml for search. In Serge Autexier, John Campbell, Julio Rubio, Volker Sorge, and Freek Suzuki, Masakazuand Wiedijk, editors, *Intelligent Computer Mathematics*, volume 5144 of *Lecture Notes in Computer Science*, pages 536–542. Springer Berlin Heidelberg, 2008.

[14] Jozef Mišutka and Leo Galamboš. System description: Egomath2 as a tool for mathematical searching on wikipedia.org. In *Proceedings of the 18th Calculemus and 10th International Conference on Intelligent Computer Mathematics*, MKM'11, pages 307–309, Berlin, Heidelberg, 2011. Springer-Verlag.

[15] NIST. Digital Library of Mathematical Functions. 4 Jan. 2012 `http://dlmf.nist.gov/`.

[16] Corneliu-Claudiu Prodescu and Michael Kohlhase. Mathwebsearch 0.5 – open formula search engine. In *Working Notes of the LWA 2011 – Learning, Knowledge, Adaptation*, 2011.

[17] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[18] C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[19] Stephen Robertson. On the history of evaluation in IR. *J. Inf. Sci.*, 34:439–456, August 2008.

[20] Gerald Salton. *The SMART retrieval system: experiments in automatic document processing*. Prentice Hall, 1971.

[21] Tefko Saracevic. Evaluation of evaluation in information retrieval. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '95, pages 138–146, New York, NY, USA, 1995. Association for Computing Machinery. `http://doi.acm.org/10.1145/215206.215351`.

[22] Petr Sojka. Exploiting Semantic Annotations in Math Information Retrieval. In Jaap Kamps, Jussi Karlgren, Peter Mika, and Vanessa Murdock, editors, *Proceedings of ESAIR 2012*, pages 15–16, Maui, USA, 2012. Association for Computing Machinery.

[23] Petr Sojka and Martin Líška. Indexing and Searching Mathematics in Digital Libraries – Architecture, Design and Scalability Issues. In James H. Davenport, William M. Farmer, Josef Urban, and Florian Rabe, editors, *Intelligent Computer Mathematics. Proceedings of 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011*, volume 6824 of *Lecture Notes in Artificial Intelligence, LNAI*, pages 228–243, Berlin, Germany, July 2011. Springer-Verlag. http://dx.doi.org/10.1007/978-3-642-22673-1_16.

[24] Petr Sojka and Martin Líška. The Art of Mathematics Retrieval. In *Proceedings of the ACM Conference on Document Engineering, DocEng 2011*, pages 57–60, Mountain View, CA, September 2011. Association for Computing Machinery. http://doi.acm.org/10.1145/2034691.2034703.

[25] Heinrich Stamerjohanns, Michael Kohlhase, Deyan Ginev, Catalin David, and Bruce Miller. Transforming Large Collections of Scientific Publications to XML. *Mathematics in Computer Science*, 3:299–307, 2010. http://dx.doi.org/10.1007/s11786-010-0024-7.

[26] Wojtek Sylwestrzak, José Borbinha, Thierry Bouche, Aleksander Nowiński, and Petr Sojka. EuDML—Towards the European Digital Mathematics Library. In Petr Sojka, editor, *Proceedings of DML 2010*, pages 11–24, Paris, France, July 2010. Masaryk University. http://dml.cz/dmlcz/702569.

[27] Abdou Youssef. Roles of math search in mathematics. In Jonathan Borwein and William Farmer, editors, *Mathematical Knowledge Management*, volume 4108 of *Lecture Notes in Computer Science*, pages 2–16. Springer Berlin / Heidelberg, 2006. 10.1007/11812289_2.

[28] Abdou Youssef. Advances in math search. *PAMM*, 7(1):1010501–1010502, 2007.

[29] Abdou Youssef. Methods of relevance ranking and hit-content generation in math search. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Towards Mechanized Mathematical Assistants*, volume 4573 of *Lecture Notes in Computer Science*, pages 393–406. Springer Berlin / Heidelberg, 2007. http://dx.doi.org/10.1007/978-3-540-73086-6_31.

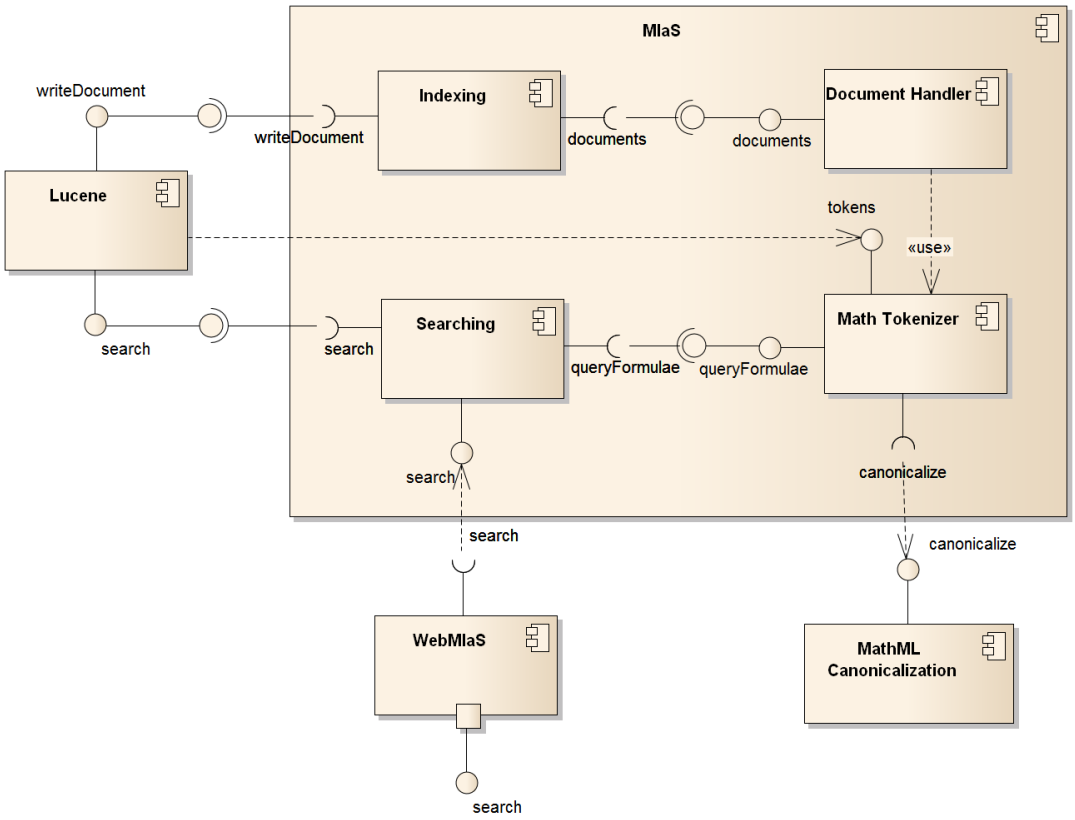
A Electronic attachments

`thesis_text.zip`	...PDF and source files of the thesis
`MIaS.zip`	...Math Indexer and Searcher
`apidocs/`	...Javadoc documentation
`example/`	...Working setup of MIaS including example documents
`src/`	...Source code
`target/`	...Built application
`pom.xml`	...Maven configuration file
`readme.txt`	...Instructions for configuring and running MIaS
`MIaSMath.zip`	...Math processing functionality of MIaS extracted as a standalone library
`apidocs/`	...Javadoc documentation
`lib/`	...Dependency libraries
`src/`	...Source code
`target/`	...Built jar archive
`pom.xml`	...Maven configuration file
`readme.txt`	...Instructions for integrating MIaSMath into Solr
`WebMIaS.zip`	...Web client for searching with MIaS
`apidocs/`	...Javadoc documentation
`src/`	...Source code
`target/`	...Built war archive
`pom.xml`	...Maven configuration file
`readme.txt`	...Instructions for configuring WebMIaS

B MIaS Design

A component structure is displayed in the diagram below. MIaS is the main application with indexing and searching capabilities. It is composed of four packages for searching, indexing, math content preprocessing and document handling. Math preprocessing uses an external library MathMLCanonicalizer. Both Searching and Indexing are clients of Lucene full-text library API. Web client WebMIaS uses MIaS API for searching and exposes a port for searching to the outside world.

Figure B.1: Web/MIaS component diagram



The class diagram (split into two diagrams for a better readability) shows the inner structure of the packages and interaction between the classes. While the first diagram omits the contents of the `cz.muni.fi.mias.math` package, the second diagram omits the other packages' contents.

Figure B.2: MIaS class diagram 1

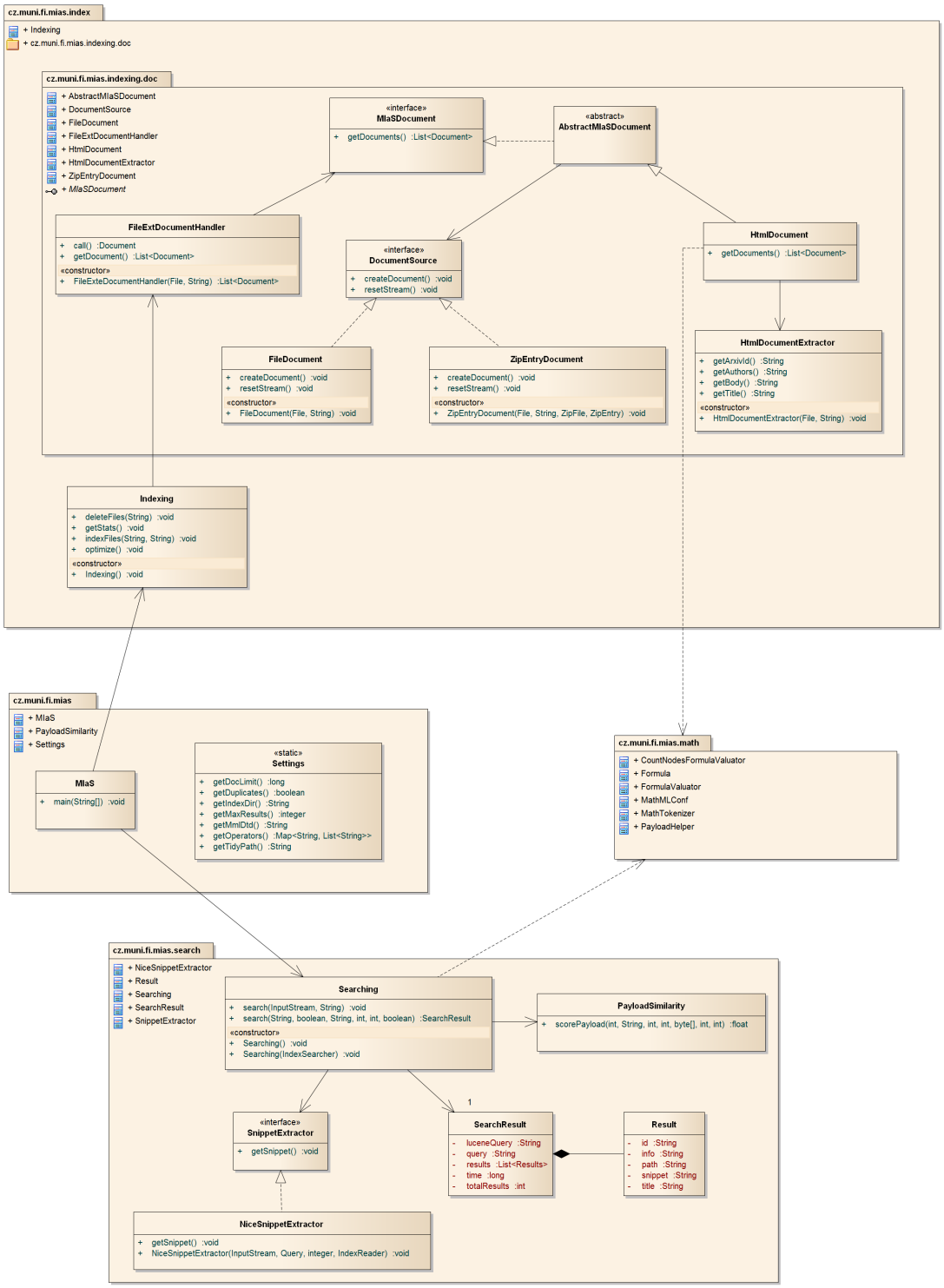
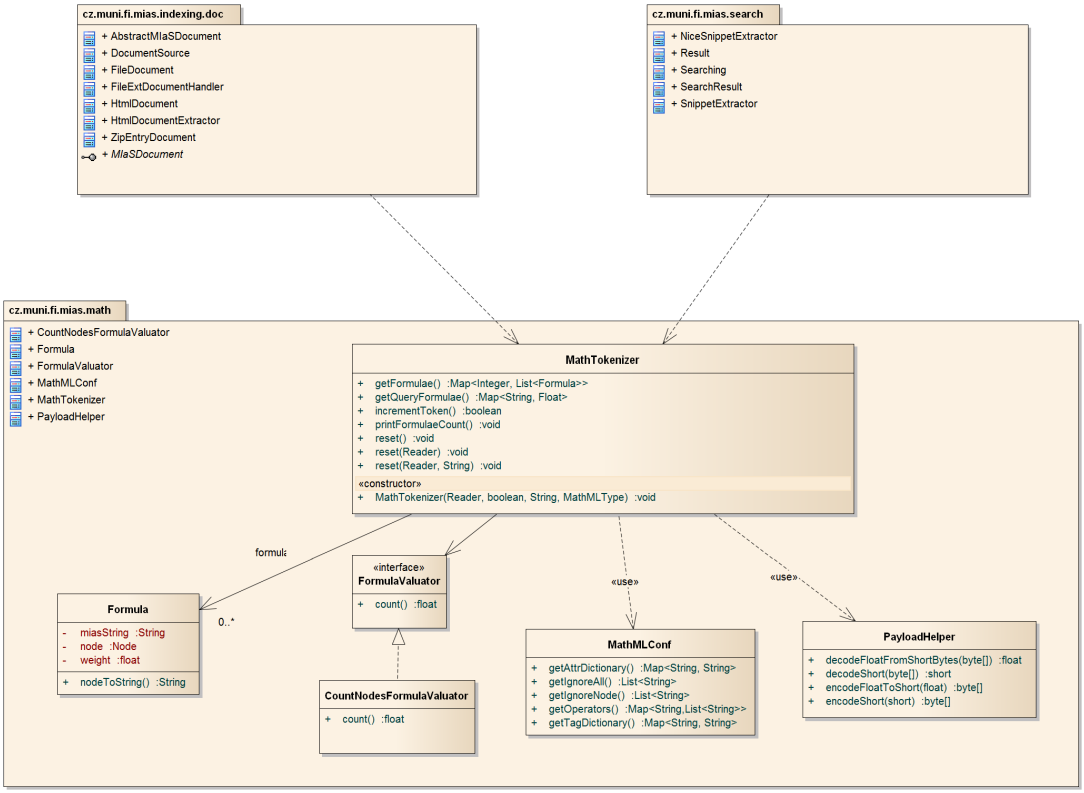


Figure B.3: MIaS class diagram 2



C MIR Happening Judges Copy

Displayed below is a verbatim copy of the document produced by the judges for the MIR Happening as it was published to the contributors after the Happening. Extracted part contains all evaluation tasks queries.

2 Evaluation Tasks

This section contains the official MIR2012 challenges, each of which has a designated article expected to be retrieved from the MIR2012 sandbox.

2.1 Formula Search (Automated)

Challenge 2.1.1. Recollect a historical formula, such as:

TEX	`\sqrt{2} = 1 + \frac{1}{3} + \frac{1}{3\dot 4} ?- \frac{1}{3\dot 4\dot 34}}`
Math	$\sqrt{2} = 1 + \frac{1}{3} + \frac{1}{34} - \frac{1}{3434}$
Example:	http://arxmliv.kwarc.info/files/1010/1010.4331/1010.4331.xhtml
Sandbox:	f005795.xhtml

But was the last operator a plus(+) or a minus(-) sign ?

Judge: Dr. Patrick Ion

Challenge 2.1.2. Retrieve instances matching:

TEX	`B_{p+n} = B_n + B_{n+1} \bmod p \ \text{for all}\ n=0,1,2,\dots`
Math	$B_{p+n} = B_n + B_{n+1} \pmod p$ for all $n = 0, 1, 2, \ldots$
Example:	http://arxmliv.kwarc.info/files/1008/1008.1573/1008.1573.xhtml
Sandbox:	f005794.xhtml

Judge: Dr. Patrick Ion

2

Challenge 2.1.3. Find examples of the use of the below metric:

TEX	`S(g) = \frac{s(g)-s_{\text{min}}}{s_{\text{max}}-s_{\text{min}}}`
Math	$S(g) = \frac{s(g)-s_{\text{min}}}{s_{\text{max}}-s_{\text{min}}}$
Example:	http://arxmliv.kwarc.info/files/1203/1203.5158/1203.5158.xhtml
Sandbox:	f005796.xhtml

Judge: Dr. Patrick Ion

Challenge 2.1.4. Find Cardy's formula:

TEX	`\frac{3\Gamma(2/3)}{\Gamma(1/3)}\eta^{1/3}\,\rule{0pt}{10pt}_2 F_1(1/3,2/3,4/3;\eta)`
Math	$\frac{3\Gamma(2/3)}{\Gamma(1/3)}\eta^{1/3}\,{}_2F_1(1/3,2/3,4/3;\eta)$
Example:	http://arxmliv.kwarc.info/files/0909/0909.4499/0909.4499.xhtml
Sandbox:	f005692.xhtml

Judge: Dr. Daniel Meyer

Challenge 2.1.5. Retrieve instances matching:

TEX	`a?x^2+b?y^2$`
Math	$ax^2 + by^2$
Example:	http://arxmliv.kwarc.info/files/0812/0812.0067/0812.0067.xhtml
Sandbox:	f004977.xhtml

Similarly for $cx^2 + dy^2$, i.e. `c?x^2+d?y^2`

Notes: This is complicated for two reasons.

- The actual variables are x_1 and x_2, not x and y (as it happens, a etc. are the same).
- We actually have $ax_1^2 + bx_2^2 + \epsilon_1 x_1 x_2$, with the possibilities of ϵ_1 being either zero or non-zero (and $cx_1^2 + dx_2^2 + \epsilon_2 x_1 x_2$ similarly).

Judge: Dr. James Davenport

Challenge 2.1.6. Retrieve instances matching:

TEX	`\frac{e^2+3}{4}2^{?l\choose 2}?n^?l`
Math	$\frac{e^2+3}{4}2^{\binom{l}{2}}n^l$
Example:	http://arxmliv.kwarc.info/files/0801/0801.2554/0801.2554.xhtml
Sandbox:	f004150.xhtml

Notes: The subtlety is that n, l are α-convertible, also called "query variables", but e is not, as it is a constant.

Judge: Dr. James Davenport

Challenge 2.1.7. Retrieve instances matching:

TEX	`?P\in \sum_{i=1}^r\Z ?{P_i}`
Math	$P \in \sum_{i=1}^{r} \mathbf{Z} P_i$
Example:	http://arxmliv.kwarc.info/files/0712/0712.3704/0712.3704.xhtml
Sandbox:	f004102.xhtml

Notes: The subtlety is that P and P_i are *independently* α-convertible, i.e. they are distinct "query variables"

Judge: Dr. James Davenport

2.2 Full-Text Search (Automated)

Challenge 2.2.1. Handle the following textual queries:

- Where can I find the formula for free cumulants in terms of the symmetric group?
- Aren't there some newer special polynomials involved?

Example:	http://arxmliv.kwarc.info/files/1010/1010.4331/1010.4331.xhtml
Sandbox:	f005795.xhtml

- Also, Kerov polynomials and zonal polynomials

Example:	http://arxmliv.kwarc.info/files/1005/1005.0316/1005.0316.xhtml
Sandbox:	f005793.xhtml

Judge: Dr. Patrick Ion

2.3 Open Information Retrieval (Semi-Automated)

Challenge 2.3.1. Retrieve instances matching:

TEX	`f_1(x_1,\ldots,x_n)<0\land f_2(x_1,\ldots,x_n)<0`
Math	$f_1(x_1,\ldots,x_n) < 0 \land f_2(x_1,\ldots,x_n) < 0$
Example:	http://arxmliv.kwarc.info/files/0801/0801.0586/0801.0586.xhtml
Sandbox:	f004115.xhtml

or conceivably: $f_1(x_1,\ldots,x_n) < 0 \land f_2(x_1,\ldots,x_n) \land \cdots \land f_m(x_1,\ldots,x_n) < 0$.

Notes: This is complicated for several reasons. The text talks about "$f_1\sigma_1 0,\ldots,f_m\sigma_m 0$", so one has to

- realise that "," is "$\land$";
- infer "$f_1(x_1,\ldots,x_n)$" from "f_1" and the earlier $f_i \in K[x_1,\ldots,x_n]$;
- infer "$f_1 < 0$" from "$f_1\sigma_1 0$" and the earlier $\sigma \in \{<,=,>\}^m$ (where $\sigma = (\sigma_1,\ldots,\sigma_m)$ is wholly implicit).

In fact, this is a remarkably hard problem, and a related question would be "what mathematically sensible queries *will* retrieve the opening paragraph of this paper?"

Judge: Dr. James Davenport

Index