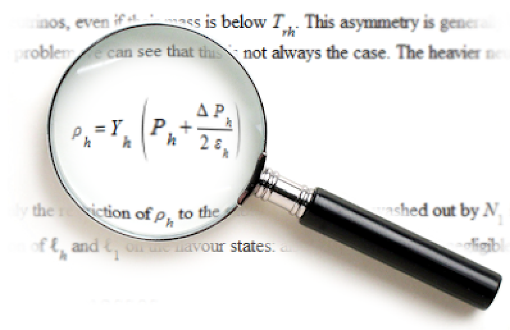
Exploiting Semantic Annotations in Math Information Retrieval

Petr Sojka, Martin Líška, Michal Růžička, David Formánek

NLP Centre, Faculty of Informatics, Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic

sojka@fi.muni.cz



Abstract

The design and architecture of MIaS (Math Indexer and Searcher), a system for mathematics retrieval is presented, and design decisions are discussed. We argue for an approach based on combining Presentation and Content MathML using: a similarity of math subformulae, semantic annotations by Mathematical Subject Classification code expansions, statistical semantics keywords generated by topic modelling (LDA), and math corpus preprocessing to disambiguate the content and find the domain collocations. The whole system is being implemented as a math-aware search engine based on the state-of-the-art system Apache Lucene. Scalability issues were checked against more than 400,000 arXiv documents with 158 million mathematical formulae. Almost three billion MathML subformulae were indexed using a Solr-compatible Lucene.

I do not seek. I find. (Pablo Picasso)

1. Introduction and Motivation

The solution to the problem of relevant, easy, and precise mathematical formulae retrieval lies at the heart of building digital mathematical libraries (DML). There have been numerous attempts to solve this problem, but none have found widespread adoption and satisfaction within the wider mathematics community. And as yet, there is no widely accepted agreement on the math search format to be used for mathematical formulae by library systems or by Google Scholar.

Computers are useless. They can only give you answers. (Pablo Picasso)

2. Approaches to Searching Mathematics

A great deal of research on has been already undertaken on searching mathematical formulae in digital libraries and on the web. The comparison of math search systems, including our new MIaS is summarized in the table below.

System	Input documents	Internal representation	Approach	α-eq.	Query language	Queries	Indexing core
MathDex	HTML, TeX/LaTeX, Word, PDF	Presentation MathML (as strings)	syntactic	×	?	text, math, mixed	Apache Lucene
LeActiveMath	OMDoc, OpenMath	OpenMath (as string)	syntactic	×	OpenMath (palette editor)	text, math, mixed	Apache Lucene
LaTeXSearch	LaTeX	LaTeX (as string)	syntactic	×	LaTeX	titles, math, DOI	?
MathWeb Search	Presentation MathML, Content MathML, OpenMath	Content MathML, OpenMath (substitution trees)	semantic	✔	QMath, LaTeX, Mathematica, Maxima, Maple, Yacas styles (palette editor)	text, math, mixed	Apache Lucene (for text only)
EgoMath	Presentation MathML, Content MathML, PDF	Presentation MathML trees (as strings)	mixed	✔	LaTeX	text, math, mixed	EgoThor
MIaS	any (well-formed) MathML	Canonical Presentation MathML trees (as compacted strings)	math tree similarity/ normalization	✔	AMS-LaTeX or MathML	text, math, mixed	Apache Lucene/ Solr

Everything you can imagine is real. (Pablo Picasso)

3. Design of MIaS

We have developed a math-aware, full-text based search engine called *MIaS* (Math Indexer and Searcher).

The top-level indexing scheme, including a detailed view of the mathematical part is shown in Figure 1.

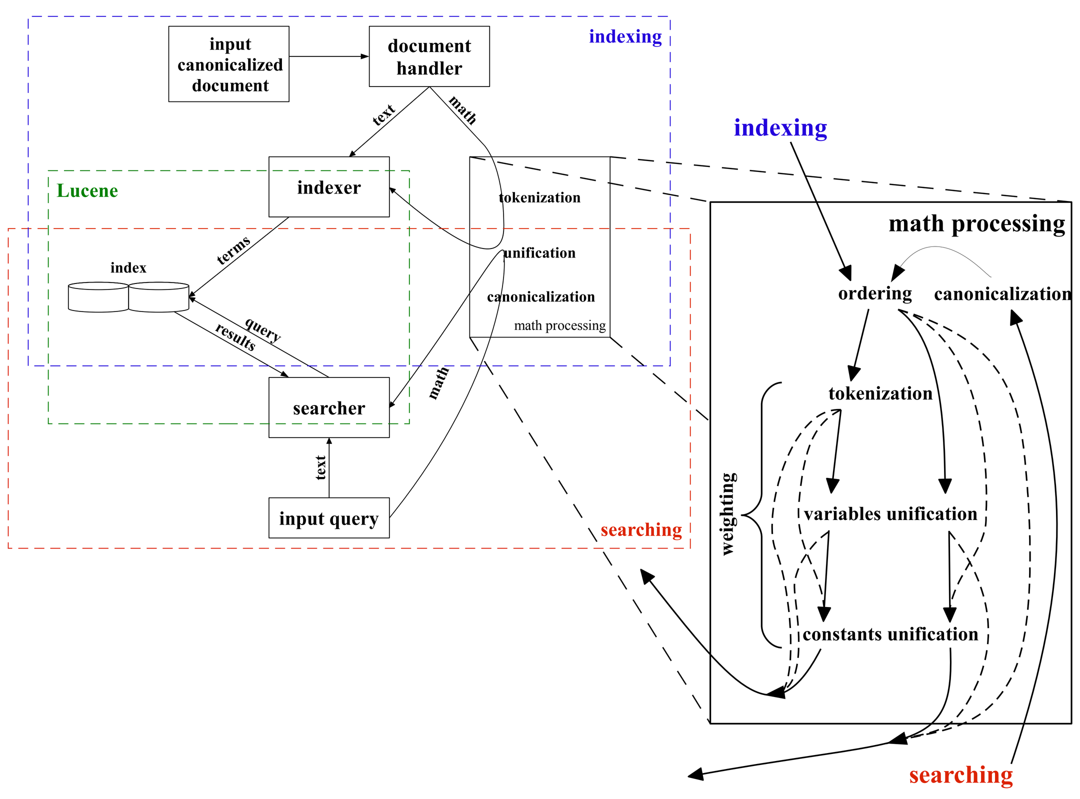


Figure 1: Scheme of the MIaS workflow of math processing

Indexing MIaS is currently able to index documents in XHTML, HTML and TXT formats. As Figure 1 shows, the input document is first split into textual and mathematical parts. The textual content is indexed in a conventional way. Mathematical expressions, on the other hand, are pre-analyzed in several steps to facilitate searches not only for exact whole formulae, but also for subparts (tokenization) and for similar expressions (formulae modifications). This addresses the issue of the static character of full-text search engines and creates several representations of each input formula all of which are indexed. Each indexed mathematical expression has a weight (relevancy score) assigned to it. It is computed throughout the whole indexing phase by individual processing steps following this basic rule of thumb—the more modified a formula and the lower the level of a subformula, the less weight is assigned to it.

At the end of all processing methods, formulae are converted from XML nodes to a compacted linear string form, which can be handled by the indexing core. Start and end XML tags are substituted by the tag name followed by an argument embraced by opening and closing parentheses. This creates abbreviated but still unambiguous representation of each XML node. For example, formula $a+b^2$, in MathML written as:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <mi>a</mi>
    <mo>+</mo>
    <msup> <mi>b</mi><mn>2</mn></msup>
  </mrow>
</math>
```

is converted to "math(mrow(mi(a)mo(+)msup(mi(b)mn(2))))" and this string is then indexed by Lucene.

Tokenization Tokenization is a straightforward process of obtaining subformulae from an input formula. MIaS makes use of Presentation MathML markup where all logical units are enclosed in XML tags which makes obtaining all subformulae a question of tree traversal. The inner representation of each formula is an XML node encapsulating all the member child nodes. This means the highest level formula—as it appears in the input document—is represented by a node named "math". All logical subparts of an input formula are obtained and passed on to modification algorithms.

Formulae Modifications MIaS performs three types of unification algorithms, the goal of which is to create several more or less generalized representations of all formulae obtained through the tokenization process. These steps allow the system to return similar matches to the user query while preserving the formula structure and α-equality.

Ordering Let us take a simple example: $a+3$ and the query $3+a$. This would not match even though it is perfectly equal. This is why a simple ordering of the operands of the commutative operations, addition and multiplication, is used. It tries to order arguments of these operations in the alphabetical order of the XML nodes denoting the operands whenever possible—it considers the priority of other relevant operators in the formula. The system applies this function to the formula being indexed as well as to the query expression. Applied to the example above, the XML node denoting variable a is named "mi", the node denoting number 3 is named "mn". "mi"<"mn" therefore $3+a$ would be exchanged for $a+3$ and would match.

Unification of Variables and Constants Let us take another example: $a+b^a$ and $x+y^x$. Again, these would not match even though the difference is only in the variables used. MIaS employs a process that unifies variables in expressions while taking bound variables into account. All variables are substituted for unified symbols (ids) in both the indexing and searching phases. Applied to the example, both expressions would unify to $\mathrm{id}_1+\mathrm{id}_2^{\mathrm{id}_1}$ and would match. This process is not applied to single symbols—this would lead to the indexing of millions of ids and searching for any symbol would end up matching all of the documents containing it.

Unification of constants is a straightforward process of substituting all the numerical constants for one unified symbol (const). This obviates the need for the exact values of constants in user queries.

Formulae Weighting During the searching phase, a query can match several terms in the index. However one match can be more important to the query than another, and the system must consider this information when scoring matched documents. For mathematical formulae the system makes use of the processing operations described above since they all produce expressions more generalized than the input ones.

Each formula has a weight attribute indexed alongside itself, which belongs to the interval $\langle 0,1)$. Weight w of the subformula contained on a certain level in a parent formula with the number of nodes (n) can be calculated in particular situations as follows:

- no changes made: $w=\frac{l^{\mathrm{level}}(1+v+c+vc)}{n}$
- unified variables: $w=\frac{l^{\mathrm{level}}(v+vc)}{n}$
- unified constants: $w=\frac{l^{\mathrm{level}}(c+vc)}{n}$
- unified both variables and constants: $w=\frac{l^{\mathrm{level}}(vc)}{n}$.

To fine tune the weighting parameters, we developed a tool with verbose output in which the behavior of the model can be observed and tested. A sample from the tool mentioned above is shown in Table 1.

We have come to the conclusion that the unification of variables interferes less with original formula meaning than the unification of number constants. For this reason, its coefficient should be higher—i.e., less discriminating. The main question then became, how discriminating the level coefficient should be. Our empirical deduction is that going deeper in a structural tree should be discriminating, the precise match on a lower level should still score more than any unified formula on the level above, as could be seen in Table 1: $\frac{1}{a+3}$ (row 5) is an exact match on the second level and its score is higher than unified expressions matched on the first level (rows 2, 3 and 4).

This led us to the valuation of level weighting coefficient $l=0.7$, unification weighting coefficient $v=0.8$ and constant weighting coefficient $c=0.5$.

In Figure 2 the whole formula preprocessing process is illustrated together with its subformulae weightings.

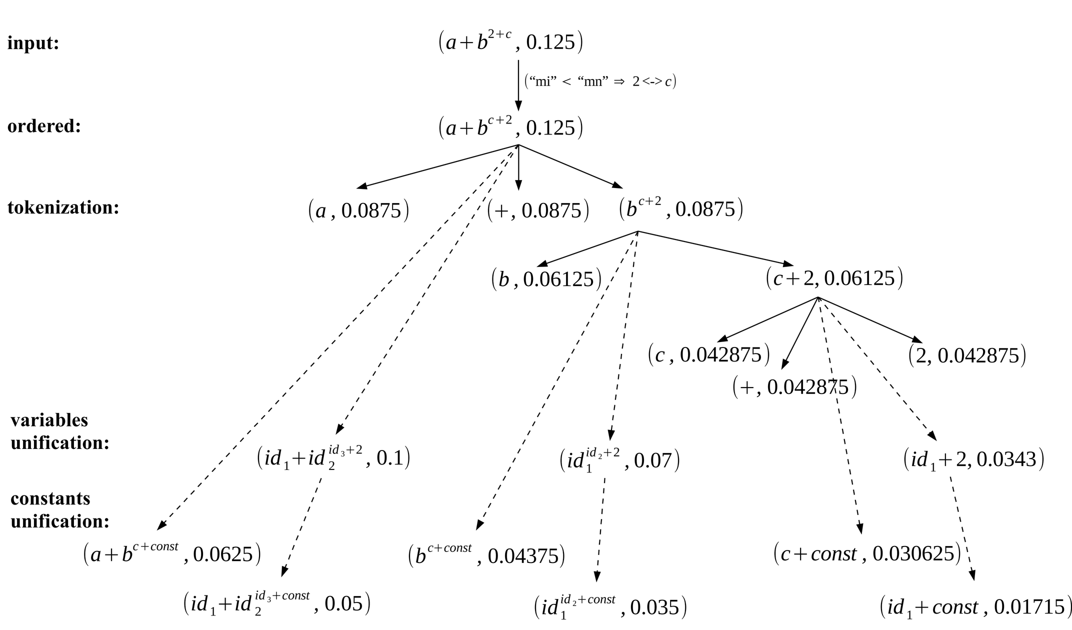


Figure 2: Example of formula preprocessing. Ordered pairs are (<expression written naturally>, <it's weight>). All expressions as shown are indexed, except for the original one.

Table 1: Example of weighting function on several formulae. Original 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=$[\mathrm{id}_1+3]$, 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=$[\mathrm{id}_1+3]$, 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=$[\mathrm{id}_1+5]$, 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=$[\mathrm{id}_1+10]$, 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=$[\frac{1}{\mathrm{id}_1+3}]$, 0.11667=$[1,a+3]$, 0.09334=$[\mathrm{id}_1+3]$, 0.08334=$[\frac{\mathrm{const}}{a+\mathrm{const}}]$, 0.08167=$[+,3,a]$, 0.06667=$[\frac{\mathrm{const}}{\mathrm{id}_1+\mathrm{const}}]$, 0.05833=$[a+\mathrm{const}]$, 0.04667=$[\mathrm{id}_1+\mathrm{const}]$	1.26	0.04667$[\mathrm{id}_1+\mathrm{const}]$ + 0.11667$[a+3]$ + 0.09334$[\mathrm{id}_1+3]$ + 0.05833$[a+\mathrm{const}]$
$\frac{1}{b+3}$	0.16667=$[\frac{1}{b+3}]$, 0.13334=$[\frac{1}{\mathrm{id}_1+3}]$, 0.11667=$[b+3,1]$, 0.09334=$[\mathrm{id}_1+3]$, 0.08334=$[\frac{\mathrm{const}}{b+\mathrm{const}}]$, 0.08167=$[b,3,+]$, 0.06667=$[\frac{\mathrm{const}}{\mathrm{id}_1+\mathrm{const}}]$, 0.05833=$[b+\mathrm{const}]$, 0.04667=$[\mathrm{id}_1+\mathrm{const}]$	0.56	0.04667$[\mathrm{id}_1+\mathrm{const}]$ + 0.09334$[\mathrm{id}_1+3]$
$\frac{1}{a+5}$	0.16667=$[\frac{1}{a+5}]$, 0.13334=$[\frac{1}{\mathrm{id}_1+5}]$, 0.11667=$[1,a+5]$, 0.09334=$[\mathrm{id}_1+5]$, 0.08334=$[\frac{\mathrm{const}}{a+\mathrm{const}}]$, 0.08167=$[a,5,+]$, 0.06667=$[\frac{\mathrm{const}}{\mathrm{id}_1+\mathrm{const}}]$, 0.05833=$[a+\mathrm{const}]$, 0.04667=$[\mathrm{id}_1+\mathrm{const}]$	0.42	0.04667$[\mathrm{id}_1+\mathrm{const}]$ + 0.05833$[a+\mathrm{const}]$
$\frac{1}{c+10}$	0.16667=$[\frac{1}{c+10}]$, 0.13334=$[\frac{1}{\mathrm{id}_1+10}]$, 0.11667=$[1,c+10]$, 0.09334=$[\mathrm{id}_1+10]$, 0.08334=$[\frac{\mathrm{const}}{c+\mathrm{const}}]$, 0.08167=$[+,c,10]$, 0.06667=$[\frac{\mathrm{const}}{\mathrm{id}_1+\mathrm{const}}]$, 0.05833=$[c+\mathrm{const}]$, 0.04667=$[\mathrm{id}_1+\mathrm{const}]$	0.19	0.04667$[\mathrm{id}_1+\mathrm{const}]$

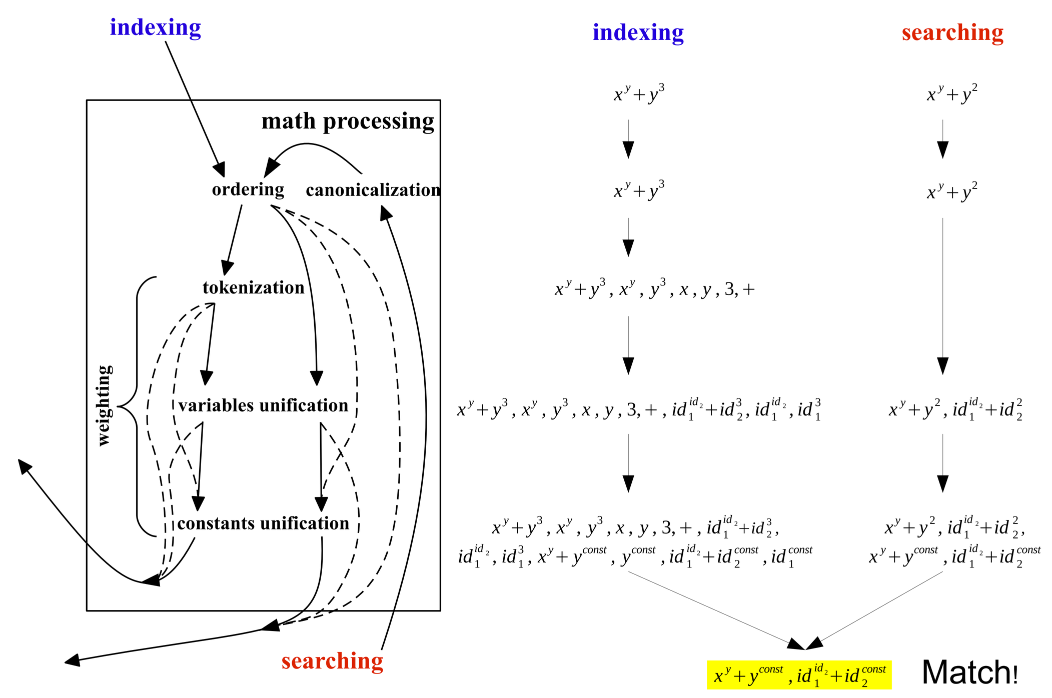


Figure 3: Math-aware search in MIaS

Searching In the search phase, user input is again split into mathematical and textual parts. Formulae are then reprocessed in the same way as in the indexing phase, except for tokenization—which we doubt that users are likely to query, for example $\frac{a+b}{c}$ wanting to find documents only with occurrences of variable c. That means the queried expressions are first ordered, then unified. This produces several representations which are connected to the final query by the logical OR operator.

A very positive value has its price in negative terms...
the genius of Einstein leads to Hiroshima. (Pablo Picasso)

4. Evaluation and Implementation

For large scale evaluation, we needed an experimental implementation and a corpus of mathematical texts. The Math Indexer and Searcher is written in Java. The role of full-text indexing and searching core is performed by Apache Lucene 3.1.0. The mathematical part of document processing can be seen as a standalone pluggable extension to any full-text library, however it would need custom integration for each one. In the case of Lucene, a custom Tokenizer (MathTokenizer) has been implemented.

For the textual content of documents, Lucene's StandardAnalyzer is employed. In MathTokenizer, TermAttributes are used for carrying strings of math expressions and PayloadAttribute for storing weights of formulae. Lucene's practical scoring function for every hit document d by query q with each query term t is as follows:

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

Corpus of Mathematical Documents MREC A document corpus MREC (version 2011.3.324) was used to evaluate the behaviour of the system we modelled. The documents come from the arXMLiv project that is converting document sets from arXiv into XHTML + MathML (both Content and Presentation) [11].

We were able to gather great amount of documents in MREC corpus version 2011.4.439 to test our indexing system. This corpus consists of 439,423 arXMLiv documents containing 158,106,118 mathematical formulae. 2,910,314,146 expressions were indexed and the resulting size of the index is 47 GB. Sizes of uncompressed and compressed corpora size are 124 GB and 15 GB, respectively. MREC corpora are available to the community for download from MREC web page http://nlp.fi.muni.cz/projekty/eudml/MREC/ so that other math indexing engines could be compared with MIaS on the same data.

Results MIaS demonstrated the ability to index and search a relatively vast corpus of real scientific documents. Its usability is highly elevated thanks to its preprocessing functions together with formulae weighting model. The ability to search for exact and similar formulae and subformulae, more so with customizable relevancy computation, demonstrates an unquestionable contribution to the whole search experience.

We have created a demo web interface WebMIaS which is publicly available on the MIaS web page http://nlp.fi.muni.cz/projekty/eudml/mias/.

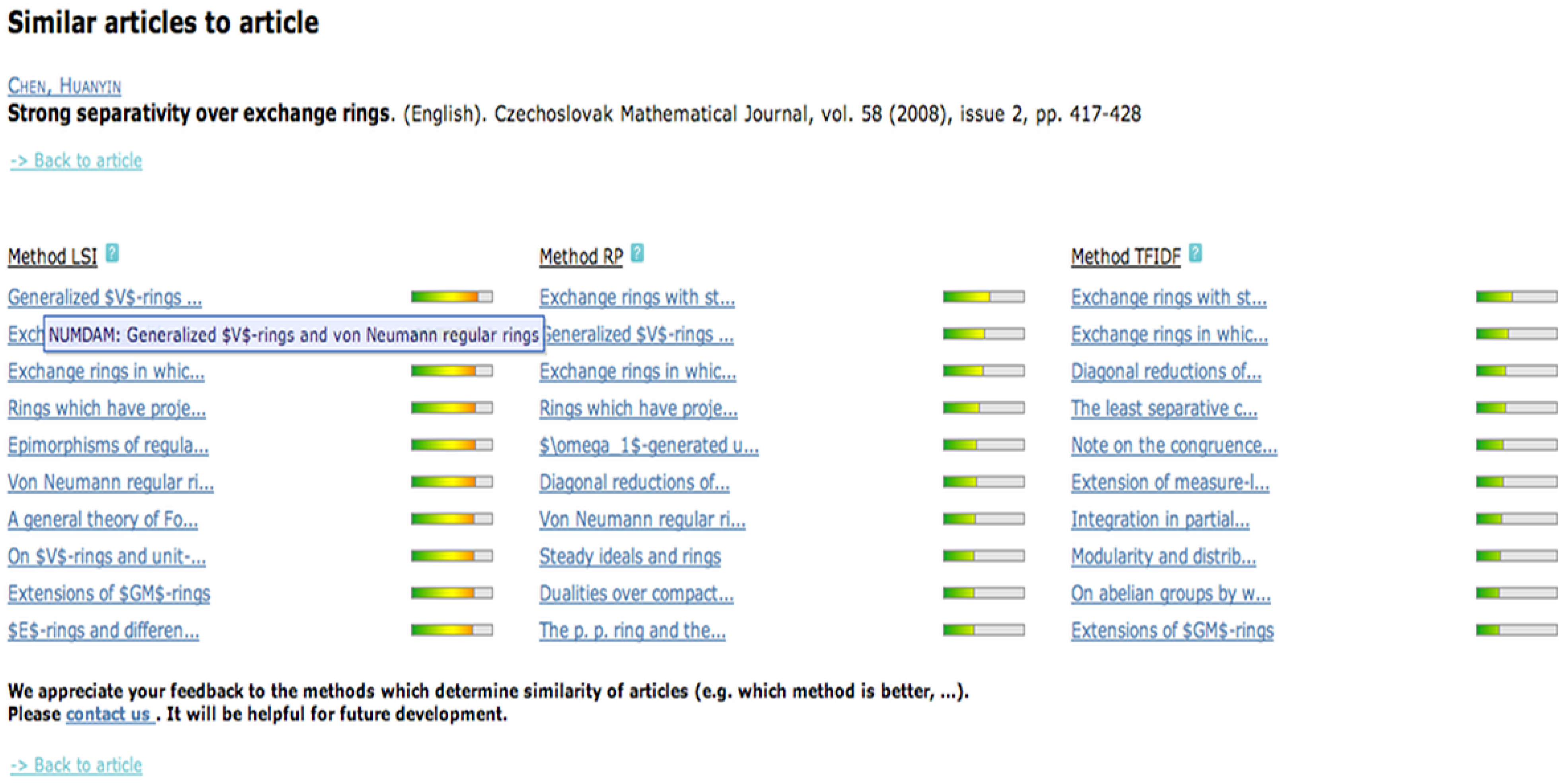
Our WebMIaS interface supports queries in two different notations—in AMS-LaTeX and MathML. Mathematical queries are additionally canonized using canonicalizer program developed primarily for semantic information retrieval to improve the query and to avoid notation flaws restraining proper results retrieval. Portability of the interface is increased by using MathJax for rendering of mathematical formulae in snippets.

Scalability Testing and Efficiency We have devised a scalability test to see how the system behaves with an increasing number of documents and formulae indexed. Subsets containing 10,000, 50,000, 200,000 and the complete 324,060 documents were gradually indexed and several values were measured: the number of input formulae, the number of indexed formulae, the indexing time and the average query time. Indexing time of this corpus was 1378.82 min, e.g. almost 23 hours.

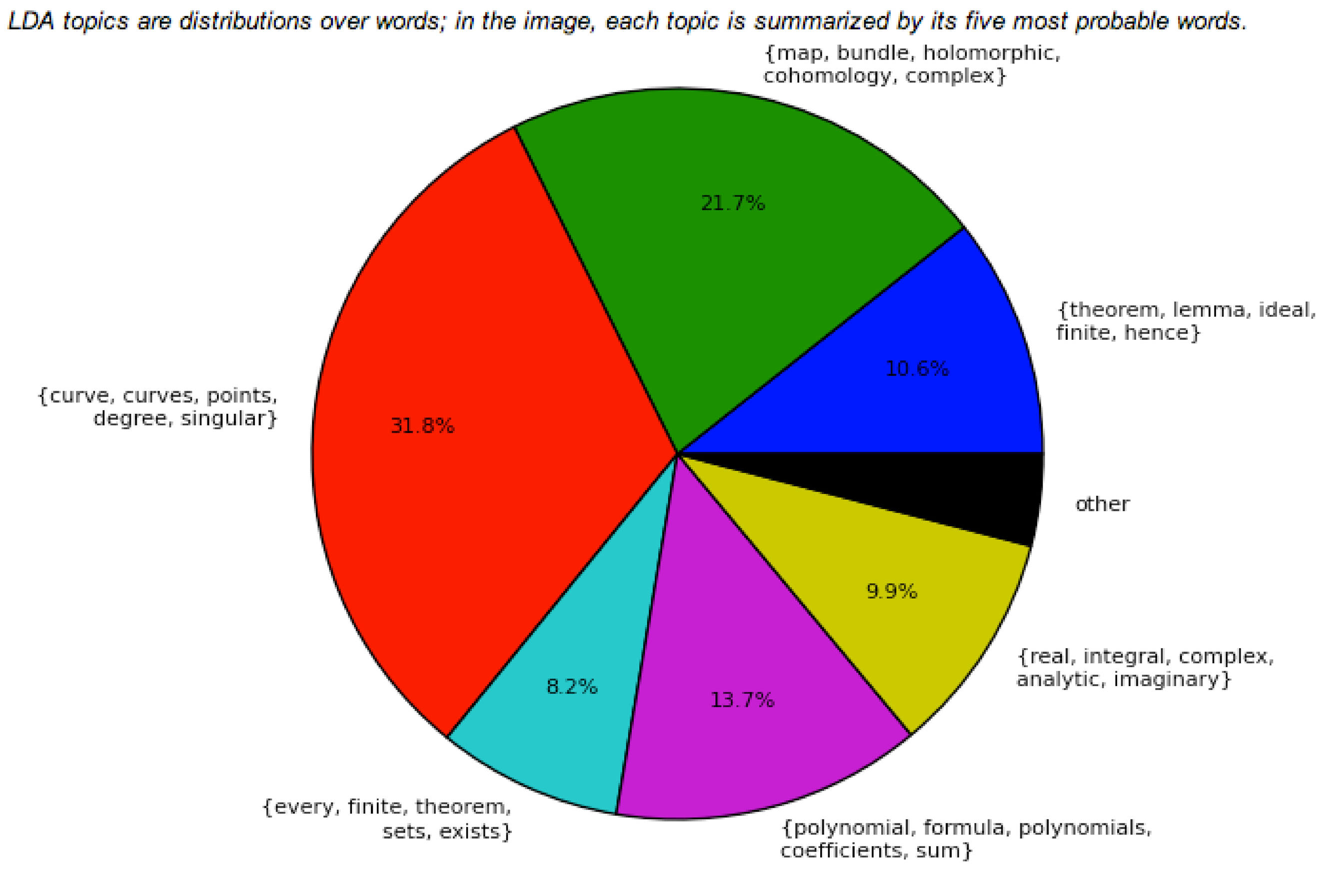
Table 2: Scalability test results (run on 32 GB RAM, quad core AMD Opteron$^{\mathrm{TM}}$ Processor 850 driven machine).

Documents	Input formulae	Indexed formulae	Indexing time [min]	Average query time [ms]
10,000	3,450,114	65,194,737	39.15	32
50,000	17,734,342	334,078,835	201.68	178
200,000	70,102,960	1,316,603,055	889.28	576
324,060	112,055,559	2,129,261,646	1,292.16	789

ESAIR 2012, Maui, Hawaii, USA, November 2, 2012, doi>10.1145/2034691.2034703



LDA Topics Pie Chart for math.0406240:
Each slice represents a different topic. The size of the slice corresponds to "how much is the article about this topic?". Topics which contribute <6% to the above document are aggregated under "other".
LDA topics are distributions over words; in the image, each topic is summarized by its five most probable words.



5. NLP for IR: Semantics for Navigational and Research Search

Natural Language Processing techniques as used in corpus management systems such as the Sketch Engine, are being used for document preprocessing (collocations, named entity recognition, word sketches, They are able to reach web scalability and avoid inference problems.
The main ideas are 1) to augment surface texts (including math formulae) with additional linked representations bearing semantic information (expanded formulae as text, canonicalized text and subformulae) for indexing, including support for indexing structural information (expressed as Content MathML or other tree structures) and 2) use semantic user preferences to order found documents.
A semantic, math-aware search is a gateway to the vast treasure of knowledge in Digital [Mathematical] Libraries (DML) as EuDML [3]. There are two main types of search: *navigational* and *research search*.
The goal of a navigational (exploratory) search is to locate documents or web pages related to the user's intention usually expressed as a sparse set of keywords. Research searches tend to be more fine-grained: their goal is to reveal again (re-search) evidence of a piece of previously published information—already known paper, theorem, lemma or equation. Both types of searches benefit from proper handling of *semantics*—that is, the meanings of *words* and their relationships. Both use cases benefit from possibility to narrow search by domain of interest specified by user as e.g. Mathematical Subject Classification (MSC) numbers.
To create a 'killer application' for EuDML—semantic, math-aware search for STEM digital libraries—we are developing the Math Indexer and Searcher (MIaS) [9, 10] and a user interface WebMIaS [6]. In its present form, MIaS primarily supports navigational searches, and it is unique in supporting not only words, but also mathematical formulae heavily used in STEM papers. To support better research searching and to improve navigational searching we are expanding our indexing terms with several types of *semantic annotations*—topical terms, canonical formulae terms and interlingual terms for multilingual retrieval.

If I don't have red, I use blue. (Pablo Picasso)

6. Problems of State of the Art

Semantic searching and the semantic web have become buzzwords today, naming different approaches to search. Google uses it for its *Knowledge Graph* of millions of interlinked words and collocations. Systems like GoPubMed build on ontology generation and usage. Wolfram Alpha believes in mathematical descriptions of a computable universe of knowledge. All are quite costly and time-consuming tasks.
Hakia, Sensebot, Powerset, DeepDyve and Cognition are further examples of 'semantic search' systems. All systems exploit context and common sense knowledge with natural language analyses of a surface form representation (bag-of-words) of documents and queries, trying to narrow the *semantic gap* between different layers of document representations: strings of optically recognized characters, words (morphology), word n-grams (collocations, phrases) to disambiguated word meanings and related topics which depend on an understanding of pragmatics. Narrative aspects of documents are usually neglected in current approaches, as is math formulae handling.
As clearly expressed by Jeff Dean (Google) in his Google I/O 2008 talk, the need for the scalability of web search demands a new generation of indexing design for every new order of magnitude of the number of documents. Semantic enhancements thus have to be computed, disambiguated and indexed in advance, limiting on-the-fly search query computations to linear or rather sublinear (*constant time*) algorithms. Costly *semantic inference* algorithms would increase search system response latency too much. Not enough time for inference should be compensated for by indexing precomputed multiple representations and *canonical semantic annotations* to increase the search system precision and performance (caching the index in distributed RAM is possible), as we are doing in MIaS.

Anything new, anything worth doing, can't be recognized. (Pablo Picasso)

7. Semantic Canonical Annotations

We believe in an empiricist approach to the natural language processing of documents, and their retrieval enhanced by semantic canonicalized annotations. NLP-based corpora management systems like Sketch Engine [5] with underlying corpus tools [2] narrow the gap between surface text and sought after meaning. Tools permit a document to appear as a list of tokenized words in uniquely numbered positions, to add part-of-speech tags, to compute collocations and phrases using various metrics (logDice, MI-score) and to create and store additional variant semantic annotations linked to document positions. MIaS allows the indexing of tree structures (as Presentation or Content MathML), in addition to standard 'bag-of-word' indexing, as a Lucene plug-in [9, 10], allowing scalable indexing of Digital Mathematical Libraries (DML).
The processing pipe starts with documents as scanned bitmaps (almost 80% in EuDML) or born-digital PDFs. Bitmaps are processed by math-aware OCR system Infty [12], and born-digital PDFs by MaxTract program [1] to get Presentation MathML.
In addition to a canonical version of the Presentation MathML formulae tree, other normalized representations, 'annotations', are created, weighted and indexed. They represent the structure of more general (sub)formulae employing variable, constant and a common subterm unification. If unambiguous or weighted Content MathML(s) can be created from Presentation MathML, it is also indexed. Word terms of formulae expanded as vocalized for reading aloud are also indexed, as interlingual parts of a document. To minimize the number of indexed entries, the process of *canonicalization* involves converting annotation into canonical representation. In MathML tree indexing, e.g., lexicographical ordering is used for normalizing math terms with commutative operators.
We are also using the Gensim tool [8] for computing weighted document LDA topics from document representation enhanced by the above-mentioned annotations, to iteratively enhance semantic annotations by adding new topical indexing terms via multilingual LDA mappings [7], and by translated set of keywords describing paper classifications (MSC is attached to virtually all math journal paper nowadays).
Document similarities are used to weigh matched terms to compute rankings to order found documents: an improvement to match user expectation is taking into account user preferences stated as subdomains of interest specified by MSC numbers. Superdocument (concatenation) of known papers of given MSCs in DL is created and similarity of this superdocument with documents in query hits (by Gensim implementation of LDA) is used to (re)order found document list: similarity and explicit hit rankings are multiplied to get a new ranking of query hits.

Art is the elimination of the unnecessary. (Pablo Picasso)

8. Canonicalization: Normalizing the Content

We have realized that the key to quality retrieval is to normalize mathematical expressions by converting the into the canonical representation. We have not found any tools to fit this goal, so we are implementing a new, modular one, in Java (using StaX for speed). The main design imperative is the modularity, simplicity, extensibility and flexibility.
Canonicalizer consists of a dozen of canonicalization modules, both for Presentation and Content MathML. We preferably use Content MathML over Presentation MathML, and eventually will convert even Presentation MathML to it in the future. Canonical representation still might be ambiguous, but new disambiguation modules are planned in the future to decrease the ambiguity and increase precision.
There are different notations possible for the same formulae (Matlab and InftyReader MathML):

```
generate::MathML(x^2 + y^2,
                 Content = FALSE, Annotation = FALSE)
<math xmlns='http://www.w3.org/1998/Math/MathML'>
  <mrow xref='No7'>
    <msup xref='No3'>
      <mi xref='No1'>x</mi>
      <mn xref='No2'>2</mn>
    </msup>
    <mo>+</mo>
    <msup xref='No6'>
      <mi xref='No4'>y</mi>
      <mn xref='No5'>2</mn>
    </msup>
  </mrow>
</math>

<math xmlns="http://www.w3.org/1998/Math/MathML">
    <msup>
        <mi mathvariant="italic">x</mi>
        <mrow>
            <mn mathvariant="normal">2</mn>
        </mrow>
    </msup>
    <mo mathvariant="normal">+</mo>
    <msup>
        <mi mathvariant="italic">y</mi>
        <mrow>
            <mn mathvariant="normal">2</mn>
        </mrow>
    </msup>
</math>
```

Here are examples of MathML transformations (Unifying Fences, <mrow> Minimizing, Sub-/Superscripts Handling, Applying Functions):

```
<mfenced open="[">
  <mi> x </mi>
  <mi> y </mi>
</mfenced>
```

```
<mrow>
  <mo> [ </mo>
  <mrow>
    <mi> x </mi>
    <mo> , </mo>
    <mi> y </mi>
  <mrow>
  <mo> ) </mo>
</mrow>
```

```
<msqrt>
  <mrow>
    <mo> - </mo>
    <mn> 1 </mn>
  </mrow>
</msqrt>
```

```
<msqrt>
    <mo> - </mo>
    <mn> 1 </mn>
</msqrt>
```

```
<msubsup>
  <mi> x </mi>
  <mn> 1 </mn>
  <mn> 2 </mn>
</msubsup>
```

```
<msup>
  <msub>
    <mi> x </mi>
    <mn> 1 </mn>
  </msub>
  <mn> 2 </mn>
</msup>
```

```
<mi> f </mi>
<mo> &#x2061; </mo>
<mrow>
  <mo> ( </mo>
  <mi> x </mi>
  <mo> ) </mo>
</mrow>
```

```
<mi> f </mi>
<mrow>
  <mo> ( </mo>
  <mi> x </mi>
  <mo> ) </mo>
</mrow>
```

I am always doing that which I can not do, in order that I may learn how to do it. (Pablo Picasso)

9. Conclusions, Credits

We have presented an approach to mathematics searching and indexing—the architecture and design of the MIaS system. The feasibility of our approach has been verified on large corpora of real mathematical papers from arXMLiv. Scalability tests have confirmed that the computing power needed for fine math similarity computations is readily available; this would allow the use of this technology for projects on a European or world-wide scale.

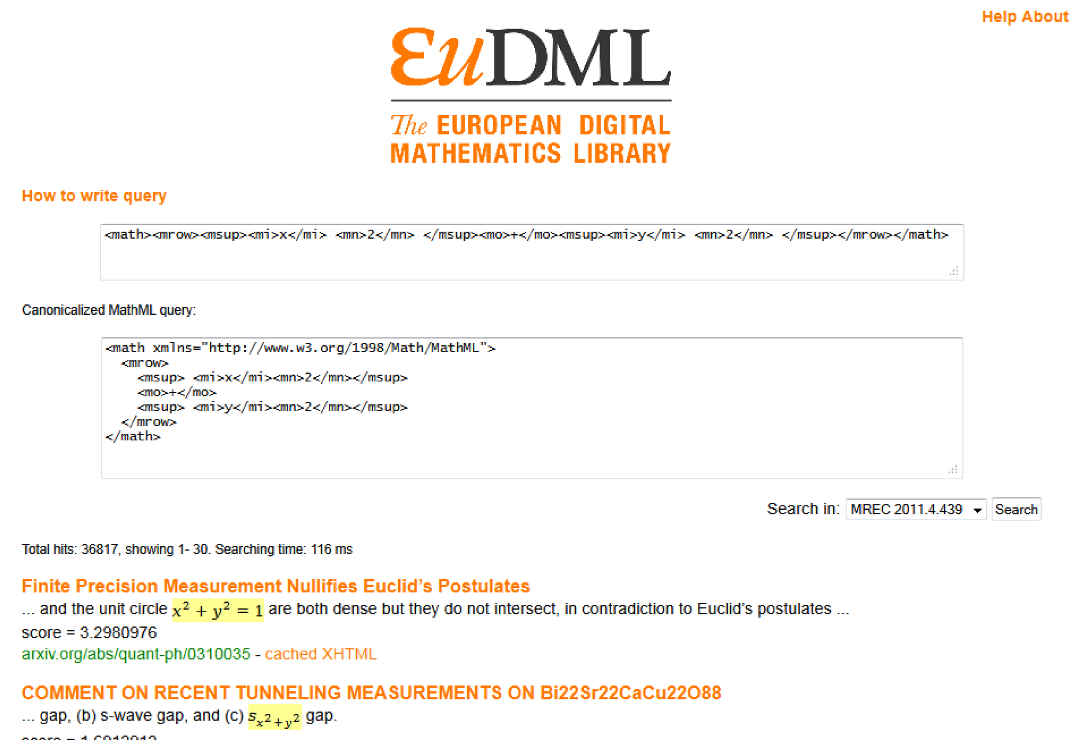


Figure 4: Web interface of MIaS for The European Digital Mathematics Library
We have also described several semantic annotations and enhancements that improve math information retrieval in DMLs. We are using the MREC corpus [6] of 438,000 preprocessed arXiv articles with 158 million mathematical formulae. for our annotation experiments and evaluation. We have found the recent paper [4] inspiring not only for suggested visual interaction with DML corpus based on topics, but also through a similar document metric that is proportional to LDA topics overlap.
Semantic annotations and techniques used during a multiple phase NLP based DML indexing workflow are promising to increase F-measure performance of WebMIaS.

Acknowledgements. Our work has been partially supported by the European Union through its Competitiveness and Innovation Programme (Policy Support Programme, "Open access to scientific information", Grant Agreement No. 250503).

References

[1] J. B. Baker, A. P. Sexton, and V. Sorge. MaxTract: Converting PDF to LaTeX, MathML and Text. In J. Jeuring, J. A. Campbell, J. Carette, G. D. Reis, P. Sojka, M. Wenzel, and V. Sorge, editors, *AISC/DML/MKM/Calculemus*, volume 7362 of *Lecture Notes in Computer Science*, pages 422–426. Springer, 2012.

[2] M. Baroni and A. Kilgarriff. Large linguistically-processed web corpora for multiple languages. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*, EACL '06, pages 87–90, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[3] J. Borbinha, T. Bouche, A. Nowiński, and P. Sojka. Project EuDML—A First Year Demonstration. In J. H. Davenport, W. M. Farmer, J. Urban, and F. 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 281–284, Berlin, Germany, July 2011. Springer-Verlag. http://dx.doi.org/10.1007/978-3-642-22673-1_21.

[4] A. J. Chaney and D. M. Blei. Visualizing topic models. In *International AAAI Conference on Social Media and Weblogs*, Department of Computer Science, Princeton University, Princeton, NJ, USA, Mar. 2012.

[5] A. Kilgarriff, P. Rychlý, P. Smrž, and D. Tugwell. The Sketch Engine. In *Proceedings of the Eleventh EURALEX International Congress*, pages 105–116, Lorient, France, 2004.

[6] M. Líška, P. Sojka, M. Růžička, and P. Mravec. Web Interface and Collection for Mathematical Retrieval: WebMIaS and MREC. In P. Sojka and T. 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.

[7] X. Ni, J.-T. Sun, J. Hu, and Z. Chen. Cross lingual text classification by mining multilingual topics from wikipedia. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 375–384, New York, NY, USA, 2011. ACM.

[8] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en, software available at http://nlp.fi.muni.cz/projekty/gensim.

[9] P. Sojka and M. Líška. Indexing and Searching Mathematics in Digital Libraries – Architecture, Design and Scalability Issues. In J. H. Davenport, W. M. Farmer, J. Urban, and F. 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.

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

[11] H. Stamerjohanns, M. Kohlhase, D. Ginev, C. David, and B. 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.

[12] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori. INFTY — An integrated OCR system for mathematical documents. In C. Vanoirbeek, C. Roisin, and E. Munson, editors, *Proceedings of ACM Symposium on Document Engineering 2003*, pages 95–104, Grenoble, France, 2003. ACM.