MathWebSearch at NTCIR-10

Michael Kohlhase
Jacobs University Bremen
m.kohlhase@jacobs-university.de

Corneliu Prodescu
Jacobs University Bremen
c.prodescu@jacobs-university.de

ABSTRACT

We present and analyze the results of the MATHWEBSEARCH system in the NTCIR-10 Math pilot task, a challenge in mathematical information retrieval. MATHWEBSEARCH is a content-based search engine that focuses on fast query answering for interactive applications. It is currently restricted to exact formula search, i.e. no similarity search and no full-text search. As the MATHWEBSEARCH system has been described elsewhere, we will only present new achievements, evaluate the results, and detail future work suggested by the task results.

Team Name

KWARC

Subtasks

Math Retrieval (English)

Keywords

NTCIR, content search, substitution tree indexing, MathML

1. INTRODUCTION

Mathematics-aware information retrieval (MIR) has often been touted as one of the "killer applications" of computer support in STEM (Science, Technology, Engineering, and Mathematics). Indeed, it has been estimated that the yearly production of published research articles in mathematics exceeds 120 thousand pages a year, the amount of internal technical documents in company archives certainly exceeds this quantity by far; being able to access the knowledge locked up in them will become a determining factor for commercial success.

To create a forum for discussing MIR the well-established NTCIR (NII Test Collection for IR Systems) Project has began a mathematics pilot task in 2013. This math task [NTM] consists of four subtasks: formula search, full-text search, open MIR, and math text understanding. The MATHWEBSEARCH system was one of the six system that participated[1] in the formula search task, which consists of 22 formula queries [Koh12] and a data set of 100 000 papers in XHTML+MathML translated by LaTeXML [Mil] from the Cornell ePrint arXiv [Sta+10].

MATHWEBSEARCH is a web service that provides low-latency answers to unification queries over content MATHML expressions [Aus+10]. The standardized format makes MATHWEBSEARCH applicable to a wide range of querying tasks – all, where formulae can be transformed into content MATHML. The low-latency makes MATHWEBSEARCH well-suited as a back-end for interactive applications, e.g. web-base formula search engines or editing support services. Unification queries form the basis of an expressive query language with well-defined semantics. As substitution instances of the original query, MATHWEBSEARCH results are highly significant, if the encoding of data set and search query are adequate – i.e. do not forget or spuriously introduce salient semantic features.

In the next two sections, we will recap unification queries and the MATHWEBSEARCH system. Section 4 presents and evaluates the results obtained by MATHWEBSEARCH in the the NTCIR-10 math pilot task. Section 5 concludes the paper with tabulation of future work planned as an answer to the lessons learnt from participating in the challenge.

Acknowledgments

The work reported here here has been partially supported by the Leibniz association under grant SAW-2012-FIZ_KA-2.

2. UNIFICATION QUERIES

Retrieval of mathematical knowledge and information via unification-based queries for content-encoded mathematical formulae is very natural. **instantiation queries**, can be used to retrieve partially remembered formulae, e.g. in searching the Zentralblatt Math [ZBMath] reviews for Hölder's inequality on an integral over the absolute value over a product of functions in Figure 1[1]. Here the query is given as \int_?a^?b |?f(x)?g(x)| dx \leq ?r in extended LaTeX (query variables marked with ?). This is transformed into MATHML (which visualizes the query variables in red). The result returned by mwsd is then visualized together with a link to the review, some meta-data, and the answer substitution, which give additional information about the result and its relation to the query. Note that instantiation queries are more expressive as a query language than e.g. regular expressions supported by some text-based search engine, since we can use variable co-occurrences to query for co-occurring subterms.

To see the full power of unification-based querying, consider a student who encounters $\int_{\mathbb{R}^2} |\sin(t)\cos(t)|dt$ and wishes to know if there are any mathematical statements (like theorems, identities, inequalities) that can be applied to it. Indeed, there are many such statements (for example Hölder's inequality) and they can be found using **generalization queries**. The idea behind answering generalization queries is that the index marks universal[2] variables in subterms as generalization targets. Hence, the search engine looks for

[1]Originally 15 teams registered for the math pilot task, but only six submitted results

[1]EDNOTE: MK: revise the figure when we have further beautifications

[2]We consider an identifier as universal if it can be instantiated with-

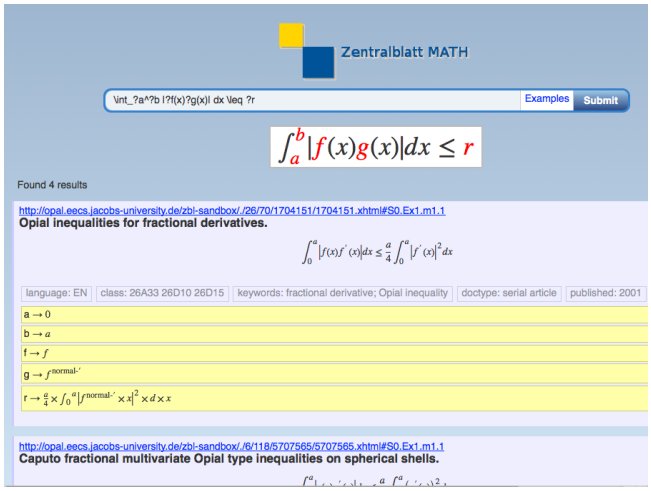

Figure 1: An Instantiation Query on [MWSZBL]

terms in the index which, after instantiating the universal identifiers, become equal to the query. For our example, we have in the index the term (we reuse the box notation for generalization targets in the index) in Figure 2, which the search engine instantiates $\boxed{x} \mapsto t, \boxed{f} \mapsto sin, \boxed{g} \mapsto cos, \boxed{D} \mapsto \mathbb{R}^2$ in order to find the generalization query. Note that the variant query $\int_{\mathbb{R}^2} |\sin(t)\cos(2t)|dt$ will not find Hölder's inequality since that would introduce inconsistent substitutions $\boxed{x} \mapsto t$ and $\boxed{x} \mapsto 2t$.

$$\int_{\boxed{D}} \left|\boxed{f}(\boxed{x})\boxed{g}(\boxed{x})\right| d\boxed{x} \leq \left(\int_{\boxed{D}} \left|\boxed{f}(\boxed{x})\right|^{\boxed{p}} d\boxed{x}\right)^{\frac{1}{\boxed{p}}} \left(\int_{\boxed{D}} \left|\boxed{g}(\boxed{x})\right|^{\boxed{q}} d\boxed{x}\right)^{\frac{1}{\boxed{q}}}$$

Figure 2: A Formula with Universal Variables in the Index

Sometimes, however, one is in the position that the searching criteria is somewhere between instantiation queries (i.e. parts are unknown) and generalization queries (parts are probably instantiated already). In this case we give the possibility to pose **unification queries** which generalize both instantiation and generalization queries. As the name suggests, the query just finds terms which are unifiable with the search expression. A query like $g^2\cos(\boxed{x}) + b\sin(\sqrt{y})$ would match the term $\boxed{a}\cos(\boxed{t}) + \boxed{b}\sin(\boxed{t})$ as we can substitute $\boxed{x} \mapsto \sqrt{y}, \boxed{t} \mapsto \sqrt{y}, \boxed{a} \mapsto g^2, \boxed{b} \mapsto b$ to get the term $g^2\cos(\sqrt{y}) + b\sin(\sqrt{y})$.

3. THE MATHWEBSEARCH SYSTEM

As the MATHWEBSEARCH system has been amply described elsewhere [KŞ06; KMP12], we will only recap enough information in order to make this paper self-contained. For NTCIR, we will focus mainly on the MATHWEBSEARCH backend mwsd (see Figure 3). To learn more about MATHWEBSEARCH-powered applications and web front-ends, see [MWSb]. The crawler and web interface components that usually connect the system to a dedicated search front-end were not needed due to the special challenge situation, where we can handle parsing and indexing of the data set and answer reporting in the NTCIR format via a simple XML parser script.

out changing the truth value of the containing expression. In formal representations like first-order logic, such variable occurrences can be effectively computed, but in semi-formal settings like mathematical textbooks, they have to be approximated by heuristic methods; see the discussion in the conclusion for details.

3.1 System Overview

mwsd in turn consists of a MATHML parser, a substitution tree index, and a term database, which maps identifiers of leaves of the index to URIs of (sub)-formula occurrences in the dataset.

To deal with large datasets, as the one provided in NTCIR-10, we developed a compressed index format. In the original index, the mapping between the next resolved term and the corresponding index node pointer was stored in red-black tree data structures. Since our main usage pattern is first index a large, static corpus and then run queries on it, we decided to switch from the rbtree maps to constant-size, sorted vectors. The main space gain comes from the fact that an extra pointer was used for each rbtree internal node. This was significant, since our payload was also a pointer (hence 50% of the space was rbtree meta-data). Furthermore, this has efficiency gains as well, since the sorted vector behaves like a perfectly balanced search tree (while this was not a guarantee for the rbtree) and memory access pattern is more cache-friendly.

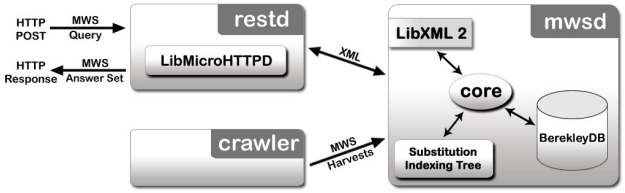

Figure 3: MWS-0.5 System Structure

3.2 Current Work

There are four main directions of development on the MATHWEBSEARCH system.

Stand-alone MATHWEBSEARCH We plan to release a version of MATHWEBSEARCH as a complete, stand-alone search engine for MathML expressions, similar to what Apache Solr supplies for text search. As described above, our system provides support for crawling, indexing and searching MathML-embedded documents. From an admin interface, the user can configure MATHWEBSEARCH to run in their own setup, indexing their own local documents or website. Configurable parameters will include the URL (or local path) from where to start indexing, the ports on which to run the RESTful query interface, as well as web search interfaces to use.

Ranking Subsystem Currently, we are using a naive hit ranking system. The only ranking heuristic in place is the depth of the substitution: expressions which match the query through more shallow substitutions are ranked higher.

We are working on more complex dynamic ranking system with complex algorithms. Infrastructure-wise, MATHWEBSEARCH is capable of retrieving up to 100,000 results in a matter of milliseconds, due to its in-memory index. On these initial results, we can run an $O(n \log n)$ ranking algorithm (remaining in millisecond range still) and cache the hits in ranked order. The exact ranking algorithm is still under development, initially it will be a combination of:

- substitution depth, as explained above
- subterm preference within the initial formula (e.g. the hit $f =?Y$ is highly relevant when searching for f)

- preference of the document corpus or MSC (Mathematics Subject Classification).

Crawlers and Corpora We are currently looking into the possibility of Building new crawlers and extending MATHWEBSEARCH to other corpora has been a constant task of our project. Newest developments include a crawler for Excel documents [KPL] [2] and a crawler for local HTML/XHTML documents.

As part of NTCIR-10, we explored the possibility of using Presentation MathML as indexing and query language. In preliminary tests, presentation queries worked reasonably well. However, in the context of NTCIR-10 queries, we failed to retrieve viable solutions using presentation MATHML. We believe this is due to the fact that Presentation MathML has no semantics in its original form and very little semantics after applying some XSLT transformations[3]. Furthermore, some constructs are inherently non-semantic: a clear example are the brackets, which appear are separate tokens (begin and end bracket) at the same level of the XML tree.

Full-Text Search Another feature we are currently developing is the integration of formula search and full-text search.

To understand the algorithm, consider the way MATHWEBSEARCH builds an index (see figure 4). Every formula ϕ is decomposed into a substitution sequence $\tilde{\phi}$ which is then inserted into a top-down substitution tree (the light gray triangular structure), every leaf in the index tree is assigned an identifier `dbid`. Then ϕ is inserted into a term database (the darker gray box) indexed with the `dbid` of $\tilde{\phi}$. Note that multiple occurrences of formulae in a document share the same `dbid`. The main idea of the TeMaSearch is to "verbalize" formulae as their `dbid` and use MATHWEBSEARCH as a form of query expansion.

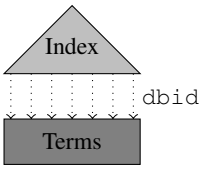

Figure 4: MATHWEBSEARCH Index

The TeMaSearch algorithm has two parts: one for indexing a document and one for querying.

At Indexing time (i.e. when we index a math document D),

I1 insert the formulae into the MATHWEBSEARCH index, remember `dbid`

I2 replace all formulae in D with their `dbid`[4] to get a document D'

I3 index D' in a bag-of-words search engine $\mathcal{S}$, e.g. Apache Solr [Solr]

An interesting question to empirically test here is how the data consumption of $\mathcal{S}$ grows with the influx of new "words" that are verbalized formulae ([KMP12] estimates the non-trivial, indexable formula occurrences of the arXiv corpus at 0.6 billion).

At query time

Q1 query Q consists of a set Q_f of formulae and a set Q_w of words.

Q2 run Q_f through MATHWEBSEARCH to get set I_f of matching `dbid`s.

Q3 run $Q' = Q_w + I_f$ through $\mathcal{S}$ to get a set R of document fragments URIs.

Q4 we return R together with the fragments of D they point to.

Note that all the wildcard handling in formulae is done by the unification algorithm at the heard of MATHWEBSEARCH and all advanced search features (wildcards in words, logical operators, etc.) can be inherited from $\mathcal{S}$.

[2] EDNOTE: CP@MK: Can you update the kwarc.bib entry with EUSPRIG as published and the KohManRab part

[3] We used XSLT transformations to highlight operators and put them in application form e.g. <m:mo>plus</m:mo> becomes <m:apply> <m:plus/> ...

[4] Note that MATHWEBSEARCH does not index subformulae natively, therefore we have to replace each formula ϕ in D with the set of `dbid`s of subformulae of ϕ

ID	hits	ID	hits	ID	hits
1	0	9	100+	17*	0
2	9	10	0	18	100+
3	21	11	0	19	0
4	4	12	0	20*	0
5	49	13*	0	21	0
6	0	14	0	22	0
7	51	15	0		
8	100+	16	0		

Table 1: Number of submitted hits

4. RESULTS & EVALUATION

The data-set provided by NTCIR consists of 100,000 XHTML+MathML documents from the arXMLiv corpus, in total 63Gbs. Our XHTML crawler generated 21 Gbs of MWS Harvest data (297 Million MATHML formulae including subexpressions) which was loaded in the MATHWEBSEARCH engine, resulting in an index occupying 10 Gb [5] of RAM and a term database of 42 Gb (on disk). Query answer times ranged from 3 to 70 milliseconds with an average of 11 ms. As one can see, the average is much lower than the maximum, as high query times were due to unusual CPU de-scheduling or I/O wait times due to other processes.

4.1 Results Analysis

In keeping with its aim to find only high-quality hits, MATHWEBSEARCH reported 434 hits, as displayed in Table 1. The items marked with $*$ contained errors (m:cerror elements) in the Content MathML query, hence 0 hits is an expected result.

The results presented above fall into one of the three categories:

a) Items which returned a significant number of hits ($100+$). These are expressions with (multiple) query variables, which have a high generality.

b) Items with few hits. These are reasonably specific queries (or highly specific in the cases of queries 2 and 4) which precise expressions.

c) Items with no hits. Out of these, some had conversion errors in the input query ($*$), some were missing from the provided dataset (not finding any hits being the expected answer), while others required some extension (unification-only matches did not exist).

4.2 Relevance

As MATHWEBSEARCH performs unification search, all hits are semantic matches. However, since the results' relevance was judged from mathematicians' perspective, semantic matches do not always suffice. The precision of our hits was rated at 18.7% fully relevant, and 33% partially relevant. While MATHWEBSEARCH topped the precision section of the competition, we aim at improving this even further through superior ranking algorithms, as discussed in Section 3.2.

4.3 Discussion

The performance profile shown by these results has its origin in the genesis of the MATHWEBSEARCH system, which came from the realization that indexing techniques from automated theorem

[5] down from ca. 23 Gb in the system reported in [KMP12]

proving could be a useful tool for MIR. The system did not report partially relevant hits as similarity-search based systems; conversely, almost all the reported results were judged as relevant.[3]

5. CONCLUSION & FUTURE WORK

We have present new developments in the MATHWEBSEARCH system and reported on the results of participating in the NTCIR-10 Math pilot task and evaluated them.

The MATHWEBSEARCH web service is open source software released under the GNU public license, the code is available from the developer portal [MWSa], search front-ends for various corpora and applications are referenced on the MATHWEBSEARCH project page [MWSc].

The performance profile shown by the NTCIR Math task has confirmed the design decisions made in the the MATHWEBSEARCH system (precision, well-defined query semantics). But it has also revealed the limitations of exactly these design decisions (inflexibility) in search practice.

We view the system reported on here as a base line for further development in four directions, which we will discuss next.

Expressive query languages We want the user to be able to describe the document fragments he/she is looking for. Extensions of unification search queries can be given in the form of *restrictions of the answer substitutions*: We can restrict the set of result instances by decorating query variables with annotations. For instance the query[6]

?f:[type=$\mathbb{R} \to \mathbb{R}$]=?G:[weight<=10]!

attributes a type constraint to the query variable f that only allows function terms as answers, and a weight constraint (only 10 constants) that limits the size the the right hand side of the result equations. Somewhat surprisingly, decorated query variables can be used for generalizations of queries as well, e.g. for specifying *ranges of (similar) numbers*: For instance, a query of the form

?x:[weight=1]^n:[st=prime(n)]+?y:[weight=1]^?n = ?z:[weight=1]^?n

could be used to describe equations similar to the the Fermat equation $x^2 + y^2 = z^2$, but restricted to prime exponents.

Similarity Search & Scoring Given a sufficiently expressive vocabulary, the query languages described above even subsume similarity search as we can describe the set of similar formulae declaratively via a schema with suitably restricted query variables. The only missing piece to a flexible similarity search regime with well-defined semantics is an equally flexible and declarative system of scoring. We conjecture that query variable restriction vocabularies for similarity search can be expressed in terms of metric relations so that we can give the intended score of the query by an arithmetic expression. For instance a query of the form

∀ ?x.?f(?x)=?y |> ed(?f,"sin")/(1+#occ(?x,?y))

uses the edit distance ed and the number of occurrences #occ (both are metrics) to compute the score on the left of the operator |>. This particular query looks for universally quantified equations whose left hand side contains a function that is similar to sin and whose right hand side may mention the quantified variables, but multiplicity is penalized.

The upshot of this treatment of similarity search is that instead of letting the MIR system decide on the notion of similarity, we believe that the user should be able to; in particular if we allow suitable front-ends to generate queries and this hide query language complexity from the end user.

Novel approaches to Ranking Currently, MATHWEBSEARCH only has a very simpleminded notion of search results ranking (size of substitutions). Ranking by a scoring regime as practiced by other MIR systems in the NTCIR may be a partial solution, but cannot distinguish top-scored results – of which there can be many in the presence of query variables. Before we can make progress here, we need to do user studies to find out what criteria mathematical practitioners have for preferring hits. A way to support this (and to delegate the problem to the user) would be to allow uses to specify a declarative ranking formula as part of the query, log these, and mine them for innovative ideas.

Interactive/Embedded Applications Web search front-ends like the one shown in Figure 1 that (essentially) provide an interface like Google does are not the only possible application, given the low turnaround time of the MATHWEBSEARCH system. We conjecture that incremental searches as they are provided by advanced text editors or the mathematical equivalence of `netspeak.org` [NSpk] would make exploring the space of sensible formulae (they are published after all) into an interactive experience. As the average query time is in range of 50 milliseconds we can send off a new query with every keystroke. Another application would be to simply monitor the number of hits for every formula as we type, a sharp drop might indicate a typo or error; and the author can be alerted in real time.

[3] EDNOTE: MK: more to say here?

[6] Here and below we use a glossed query syntax instead of extended MATHML for readability.

References

[Aus+10] Ron Ausbrooks et al. *Mathematical Markup Language (MathML) Version 3.0*. W3C Recommendation. World Wide Web Consortium (W3C), 2010. URL: `http://www.w3.org/TR/MathML3`.

[KMP12] Michael Kohlhase, Bogdan A. Matican, and Corneliu C. Prodescu. "MathWebSearch 0.5 – Scaling an Open Formula Search Engine". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (CICM) (Bremen, Germany, July 9–14, 2012). Ed. by Johan Jeuring et al. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 342–357. ISBN: 978-3-642-31373-8. URL: `http://kwarc.info/kohlhase/papers/aisc12-mws.pdf`.

[Koh12] Michael Kohlhase. *Topics for the NTCIR-10 Math Task; Math Retrieval Subtask*. Tech. rep. NTCIR, 2012. URL: `https://kwarc.info/kohlhase/papers/NTCIR10-topics.pdf`.

[KPL] Michael Kohlhase, Corneliu Prodescu, and Christian Liguda. "XLSearch: A Search Engine for Spreadsheets". submitted. URL: `http://kwarc.info/kohlhase/papers/eusprig13-xlsearch.pdf`.

[KŞ06] Michael Kohlhase and Ioan Şucan. "A Search Engine for Mathematical Formulae". In: *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2006*. Ed. by Tetsuo Ida, Jacques Calmet, and Dongming Wang. LNAI 4120. Springer Verlag, 2006, pp. 241–253. URL: `http://kwarc.info/kohlhase/papers/aisc06.pdf`.

[Mil] Bruce Miller. *LaTeXML: A LaTeX to XML Converter*. URL: `http://dlmf.nist.gov/LaTeXML/` (visited on 03/12/2013).

[MWSa] *Math Web Search*. URL: https://trac.mathweb.org/MWS/ (visited on 01/08/2011).

[MWSb] *MathWebSearch Searching Mathematics on the Web*. URL: http://search.mathweb.org (visited on 04/26/2013).

[MWSc] *MathWebSearch Searching Mathematics on the Web*. URL: http://search.mathweb.org (visited on 04/26/2013).

[MWSd] *MathWebSearch Searching Mathematics on the Web*. URL: http://search.mathweb.org (visited on 04/26/2013).

[MWSZBL] *MathWeb Search: Zentralblatt Math*. URL: http://earch.mathweb.org/zbl/ (visited on 04/21/2013).

[NSpk] *Netspeak - One words word leads to another*. URL: http://www.netspeak.org/ (visited on 04/17/2013).

[NTM] *NTICR Pilot Task: Math Task*. URL: http://ntcir-math.nii.ac.jp/ (visited on 11/11/2012).

[Solr] Apache Software Foundation. *Apache Solr*. URL: http://lucene.apache.org/solr/ (visited on 06/24/2010).

[Sta+10] Heinrich Stamerjohanns et al. "Transforming large collections of scientific publications to XML". In: *Mathematics in Computer Science* 3.3 (2010): *Special Issue on Authoring, Digitalization and Management of Mathematical Knowledge*. Ed. by Serge Autexier, Petr Sojka, and Masakazu Suzuki, pp. 299–307. URL: http://kwarc.info/kohlhase/papers/mcs10.pdf.

[ZBMath] *Zentralblatt MATH*. URL: http://www.zentralblatt-math.org/zbmath/ (visited on 06/12/2012).