System Description: MathWebSearch 0.3, A Semantic Search Engine

Michael Kohlhase[1] and Ioan A. Şucan[2]

[1] Computer Science, Jacobs University Bremen
m.kohlhase@iu-bremen.de
[2] Computer Science, Rice University,
isucan@rice.edu

Abstract. We present a search engine for mathematical formulae. The MathWebSearch system harvests the web for content representations of formulae and indexes them with substitution tree indexing. In version 0.3 we have parallelized and distributed the search server, which can be queried via a web interface or directly via a socket interface. Our experiments show that this architecture results in a scalable application.

1 Introduction

As the world of information technology grows, being able to quickly search data of interest becomes one of the most important tasks in any kind of environment, be it academic or not. We present the MathWebSearch system a search engine that can can retrieve mathematical formulae not just raw text (as a standard engine like Google can). MathWebSearch addresses the problem of searching mathematical formulae from a semantic point of view, it finds them by their structure and meaning not via their presentation.

In [KS06] we have presented the motivation, query language, and web front end of MathWebSearch 0.2. Here we describe MathWebSearch0 .3, which features significant efficiency gains and management features. The MathWebSearch system (see [Mat07] for details) is released under the Gnu General Public License [FSF91]. A running prototype of MathWebSearch 0.3 is available for testing at `http://betasearch.mathweb.org`.

A standard use case for MathWebSearch is that of an engineer trying to solve a mathematical problem, such as finding the power of a given signal $s(t)$, but only remembers that a signal's power involves integrating its square[3]. In particular he has forgotten the details of how to compute the necessary integrals. He will therefore call up MathWebSearch to search for something that looks like $\int_?^? s^2(t)dt$. MathWebSearch finds a document about Parseval's Theorem,

[3] We use this simple example mainly for expository purposes here. Other applications include the retrieval of equations that allow to transform a formula, of lemmata to simplify a proof goal, or to find mathematical theories that can be re-used in a given context (see [Nor06a] for a discussion of the latter).

more specifically $\frac{1}{T}\int_0^T s^2(t)dt = \Sigma_{k=-\infty}^{\infty}|c_k|^2$, where c_k are the Fourier coefficients of the signal. In short, even though our engineer had missed the factor in front and the integration limits, he fairly easily found the exact formula he was looking for and moreover a theorem he may be able to use.

2 The MathWebSearch System

The MathWebSearch system is a distributed application consisting of

Search Nodes that run search servers, index builders, and web crawlers. Some of the nodes contain "meta-servers" that act as gateways to others; they forward queries to a specified set of nodes and merge the received results. Thus the collection of search nodes is organized as a tree for efficient query distribution when using a large number of nodes. In case of failure of a node, the only effect is that the results that would have been produced by that node are not received by the web server.

Database Servers that store the indexed documents here realized in MySQL.

A Web Server to Communicate with Browser Clients that combines search results from the root meta-server with the documents from the databases.

An Admin Server that allows to assign the different tasks to the available nodes, or to add automatic load balancing if needed. The admin server also monitors the search nodes for node failures and re-directs search.

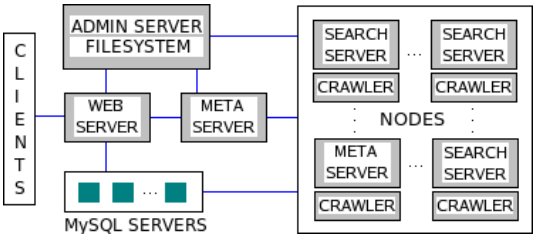

Fig. 1. System Architecture

A search server is able to run multi-threaded searches. In the age of multi-core CPUs, this becomes an important feature. The number of threads used in the search is chosen at the index building step. By construction, the theoretical speedup is linear. However, depending on the query, one of the threads may have to find all results and then no speedup is achieved. If the threads have balanced amount of work the achieved speedup is linear – or even super-linear due to cache effects. To avoid memory allocations and cache misses, the same area of memory – a memory pool – is being used in different steps of the search.

We have tested our implementation on `http://functions.wolfram.com`, which contains 90,000 mathematical formulae and yields an index with 1.5 million subterms (memory footprint 250MB). Typical query execution times are in the range of milliseconds. The search in our running example takes about 1 ms for instance. Performed experiments indicate that index sizes have little effect on search times, even for more complex searches.

The distributed architecture and multi-threaded implementation of the search nodes is the main contribution of MathWebSearch 0.3 over the previous version presented in [KS06]. It leads to significant efficiency gains (reducing mem-

ory footprint to a third and speeding up searches by an order of magnitude) and makes the system scalable to larger indexes.

Input Processing MathWebSearch can process any Xml-based content representation of mathematical formulae: MathML [W3M+03] and OpenMath [BCC+04] are supported directly, other formats e.g. Wolfram Research's Mathematica are supported if there is a converter to them.

Consider the example on the right: We have the standard mathematical notation of an equation (1), its Content MathML representation (2), and the term we extract for indexing (3). As previously stated, any mathematical construct can be represented in a similar fashion. See [KS06] for more details and syntax of search queries.

(1) Mathematical expression: $f(x) = y$

(3) Term representation: $eq(f(x), y)$

(2) Content MathML:

```
<apply><eq/>
  <apply>
    <ci>f</ci>
    <ci>x</ci>
  </apply>
  <ci>y</ci>
</apply>
```

Search modulo α-renaming becomes available via a very simple input processing trick: during input processing, we replace bound variables by generic terms. Therefore in our running example the query variable t is made generic; the query would then also find the variant $\frac{1}{T}\int_0^T s^2(x)dx = \Sigma_{k=-\infty}^{\infty}|c_k|^2$; as t is generic it could principally match any term in the index, but given the MathML constraints on the occurrences of bound variables, it will in reality only match variables (thus directly implementing α-equivalence).

An extensible framework for approximate search is also provided. For example, the user can easily perform a search for $eq(x, 5)$ but specify that numbers within range of 1 also match. $eq(x, 4)$ or $eq(x, 6)$ will then also be considered results. With the same extension we can catch typos like $fibonaci(x)$ and return $fibonacci(x)$ as well.

An addition to the current implementation is a search plugin that allows using MathWebSearch with the Firefox 2 or Internet Explorer 7 search bar. The input in this case is the term representation (called "string representation" in [KS06] and documented there); this includes the internal representation of terms (prefix notation) and any expression that Mathematica is able to parse. To require terms to be only interpreted as prefix notation, prepend a `#` to them (for example `#eq(id(x),@value)`).

Indexing Mathematical Formulae For indexing mathematical formulae on the web, we will interpret them as first-order terms and index them with a form of tree-based indexing derived from *substitution-tree indexing*.

The main difference between our technique and substitution-tree indexing is that we store the actual terms at each node instead of substitutions. This is not a space problem as we use a shared term representation. Internal nodes of the tree are *generic terms* (represented as `@string`) and represent similarities between terms. The indexed terms will appear as leaves; for any step taken down the tree,

at least one of the generic terms is replaced by a non-generic term (standard terms); when reaching a leaf, all generic terms will have been substituted. When inserting in the index, focus is placed on deepening the tree with the sole constraint of not having an empty substitution. 5terms shows a typical index for the terms $h(f(z,a,z))$, x, $g(f(z,y,a))$, $g(f(7,z,a))$, and $g(f(7,z,f))$. For clarity we also present the substitutions in the node. The subterms `@integer` are the generic subterms; for details see [KS06]. To each of the indexed terms we attach an identifier that relates the term to its XPointer [GMMW03] reference and other relevant data (see below).

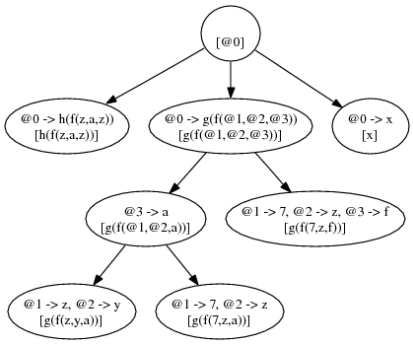

Fig. 2. An Index with Five Terms

When building indexes for multi-threaded search, we in fact build multiple indexes, equal to the number of desired threads. This is transparent to the application using the index data-structure and it is guaranteed that the different threads will have approximately equal numbers of terms in their indexes. This is needed for having approximately equal search times the search threads and thus limit the synchronization time.

Reporting Results For a search engine for mathematical formulae we need to augment the set of result items (usually page title, description, and page link) reported to the user for each hit. As typical pages contain multiple formulae, we need to report the exact occurrence of the hit in the page. We do this by supplying an XPointer reference where possible. Concretely, we group all occurrences into one page item that can be expanded on demand. Within this we order the groups by number of contained references.

For any given result a detailed view is available. This view shows the exact term that was matched and the used substitution (a mapping from the query variables specified by generic terms) to match the found result. A more serious problem comes from the fact that — as mentioned above — content representations are often the source from which presentations are generated. If MathWebSearch can find out the correspondence between content and presentation documents, it will report both to the user. Wherever possible we present two links as results: one is the *source link*, a link to the document we actually index, and the *default link*, a link to the more esthetically pleasing presentation document.

3 Conclusions and Future Work

We have presented a scalable search engine for mathematical formulae on the Internet. In contrast to other approaches, MathWebSearch uses the full content structure of formulae, and is easily extensible to other content formats. We will continue developing MathWebSearch into a production system.

A current weakness of the system is that it can only search for formulae that match the query terms up to α-equivalence or some previously defined approximation like small edit distance. Many applications would benefit from stronger equalities for instance. Our search in the running example might be used to find a useful identity for $\int_{\infty}^{0} f(x) \cdot g(x)dx$, if we know that $s(x) \cdot s(x) = s^2(x)$. MathWebSearch can be extended to a *E-Retrieval* engine (i.e. search modulo a an equational theory E or logical equivalence) without compromising efficiency by simply E-normalizing index and query terms.

The question of ranking search results for formula queries is largely unexplored territory. We are currently exploring several approaches: substitution size, familiarity of substituted constants, formula-class (prefer definition/theorem/...), formula-rank (prefer formulae that are frequently re-used/referenced).

In the long run we plan to extend MathWebSearch, so that it can take document context information into account, e.g. keywords from the text around the formulae or the topology of theories in the OMDoc format [Koh06]: It would be very useful, if we could restrict searches to formulae that are consistent with current (mathematical) assumptions.

Finally we would like to allow specification of content queries using more largely known formats, like LaTeX: strings like `\frac{1}{x^2}` or `1/x^2` could be processed as well. We are currently working on translating the ca. 400.000 TeX/LaTeX articles on Physics, Mathematics, and Computer Science in the Cornell ePrint archive (see `http://www.arXiv.org`) to content MathML, to make MathWebSearch accessible for a larger group of users. We estimate that this collection contains in the order of 10^8 non-trivial formulae, making this collection a real scalability challenge for MathWebSearch.

References

[BCC+04] Stephen Buswell, Olga Caprotti, David P. Carlisle, Michael C. Dewar, Marc Gaetano, and Michael Kohlhase. The OpenMath Standard, Version 2.0. Technical report, The Open Math Society, 2004.

[FSF91] Free Software Foundation. Gnu general public license. Software License available at `http://www.gnu.org/copyleft/gpl.html`, 1991.

[GMMW03] Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. XPointer framework. W3C Recommendation, World Wide Web Consortium, 2003.

[ICW06] Tetsuo Ida, Jacques Calmet, and Dongming Wang, editors. *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2006*, number 4120 in LNAI. Springer Verlag, 2006.

[Koh06] Michael Kohlhase. OMDoc *An open markup format for mathematical documents (Version 1.2)*. Number 4180 in LNAI. Springer Verlag, 2006.

[KS06] Michael Kohlhase and Ioan Şucan. A search engine for mathematical formulae. In Ida et al. [ICW06], pages 241–253.

[Mat07] Math web search. Web page at `http://kwarc.info/projects/mws/`.

[Nor06a] Immanuel Normann. Enhanced theorem reuse by partial theory inclusions. In Ida et al. [ICW06].

[W3M+03] The W3C Math WG. Mathematical Markup Language (MathML) version 2.0 (second edition). W3C recommendation, World Wide Web Consortium, 2003. Available at http://www.w3.org/TR/MathML2.