Faceted Search for Mathematics

by

Radu Hambasan

Bachelor Thesis in Computer Science

Prof. Michael Kohlhase
Supervisor

Date of Submission: March 7, 2015

Jacobs University — School of Engineering and Science

With my signature, I certify that this thesis has been written by me using only the indicated resources and materials. Where I have presented data and results, the data and results are complete, genuine, and have been obtained by me unless otherwise acknowledged; where my results derive from computer programs, these computer programs have been written by me unless otherwise acknowledged. I further confirm that this thesis has not been submitted, either in part or as a whole, for any other academic degree at this or another institution.

Radu Hambasan Bremen, March 7, 2015

Abstract

Faceted search represents one of the most practical ways to browse a large corpus of information. Information is categorized automatically for a given query and the user is given the opportunity to further refine his/her query. Many search engines offer a powerful faceted search engine, but only on the textual level. Faceted Search in the context of Math Search is still unexplored territory.

Advanced formula search is the desirable approach for browsing a corpus of mathematical formulae, where purely textual search would fail. MathWebSearch is such a formula search engine, combining both math and text queries. However, it does not yet provide any functionality for faceted search.

In this thesis, I solve the faceted search problem in math, by extracting formula schemata from a given set of formulae. Furthermore, I describe possible ways of integrating this with existing services.

Contents

1 Introduction

The size of digital data has been growing tremendously since the invention of the Internet. Today, the ability to quickly search for relevant information in the vast amount of knowledge available is essential in all domains. As a consequence, search engines have become the prevalent tool for exploring digital data.

Although text search engines (e.g. Google [4] or Bing [2]) seem to be successful for the average user, they are limited when it comes to finding scientific content. This is because STEM[1] documents are mostly relevant for the mathematical formulae they contain and math cannot be properly indexed by a textual search engine, because the hierarchical structure of the content is also important.

A good math search engine is therefore needed in several applications. For example, a large airline may have many ongoing research projects and could significantly improve efficiency if they had a way of searching for formulae in a corpus containing all their previous work. The same holds for all large physics-oriented research centers, such as CERN. Valuable time would be saved if scientists would have a fast, reliable and powerful math search engine to analyse previous related work. As a third application, university students should be mentioned. Their homework, research and overall study process would be facilitated once they are provided with more than textual search. For all these applications, we need first a strong math search engine and second a large corpus of math to index.

The Cornell e-Print Archive, ArXiv, is an example of such a corpus, containing over a million STEM documents from various scientific fields (Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance and Statistics) [1]. Having almost a million documents, with possibly more than a billion formulae, the search engine must provide an expressive query language and query-refining options to be able to retrieve useful information. One service that provides both these is the Zentralblatt Math service [11].

Zentralblatt Math now employs formula search for access to mathematical reviews [6]. Their database contains over 3 million abstract reviews spanning all areas of mathematics. To explore this database they provide a powerful search engine called “structured search”. This engine is also capable of faceted search. Figure 1 shows a typical situation: a user

[1] Science, Technology, Engineering and Mathematics

searched for a keyword (here an author name) and the faceted search generated links for search refinements (the **facets**) on the right. Currently, facets for the primary search dimensions are generated – authors, journals, MSC, but not for formulae. In this way, the user is given the ability to further explore the result space, without knowing in advance the specifics of what he/she is looking for. Recently, formula search has been added as a component to the structured search facility. However, there is still no possiblity of faceted search on the math content of the documents.

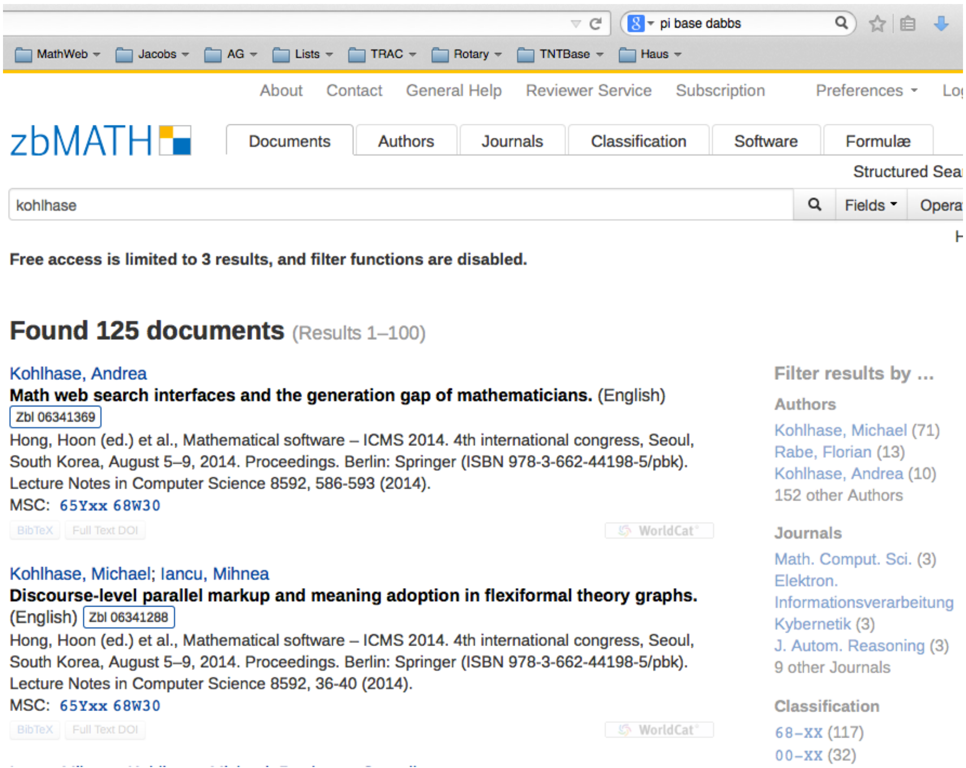

Figure 1: Faceted Search in ZBMath

There are multiple ways in which we could understand a "math facet". One way would be through the MSC classification [8]. However, this would be rather vague because it will only provide info about the field of mathematics to which an article belongs. If the authors use formulae from another field in their paper, the results will suffer a drop in relevance.

I solved this problem by extracting formula schemata from the query hits as formula facets. A math facet consists of a set of formula schemata generated to further disambiguate the query by refining it in a new dimension. For instance, for the query above we could have the formulae in Figure 2, which allows the author to drill in on *i*) variation theory and minimal surfaces, *ii*) higher-order unification, and *iii*) type theory. Following the MWS (see 2.2) tradition, the red identifiers stand for query variables, their presence making the results **formula schemata**.

These formula schemata were manually created, but for an application we need to generate them automatically from the query. This is the algorithmic problem I explore in the thesis.

$\int_M \Phi(d_p f) dvol$

$\lambda X.h(H^1 X) \cdots H^n X$

$\frac{\Gamma \vdash A \gg \alpha}{D}$

Figure 2: formula facets

2 Preliminaries

In this section we describe the existent systems on which our work will be based, with the intention of making this thesis self-contained. We will present these systems in detail in the rest of this section, but below is a summary of the role they play in our work:

- **MathWebSearch** which provides the necessary index structure for schema search.
- **Elasticsearch** which provides hits in response to text query, as well as run aggregations on the hits. These hits represent formulae to be schematized.
- **Zentralblatt Math** which also provides formulae to be schematized. The algorithm to obtain these hits, corresponds to their current implementation of faceted search.

2.1 Project Goals

[1] EdN:1

2.2 MathWebSearch

At its core, the MathWebSearch system (MWS) is a content-based search engine for mathematical formulae. It indexes MathML formulae, using a technique derived from automated theorem proving: Substitution Tree

[1] EdNote: TODO

Indexing. Recently, it was augumented with full-text search capabilities, combining keywords query with unification-based formula search. The engine serving text queries is Elasticsearch 2.3. From now on, in order to avoid confusion, we will refer to the core system (providing just formula query capability) as MWS and to the complete service (MWS + Elasticsearch) as TeMaSearch.

The overall workflow of TeMaSearch is the following:

1. HTML5 documents representing mathematical articles are crawled to generate MWS harvests [9]. .harvest is an extension of MathML which MWS can index. Its role is to separate the math from the text in a given document.

2. MWS indexes the harvests.

3. a second pass is made over the harvests to generate annotated documents (see below).

4. Elasticsearch indexes the annotated documents.

5. Everything is now ready for answering queries. When a query is issued, MWS will answer the mathematical part and Elasticsearch will answer the text part. The results are combined through a NodeJS [10] proxy to send a final result set.

Each mathematical expression is encoded as a set of substitutions based on a depth-first traversal of its Content MathML tree. Furthermore, each tag from the Content MathML tree is encoded as a TokenID, to lower the size of the resulting index. The (bijective) mapping is also stored together with the index and is needed to reconstruct the original formula. The index itself is an in-memory trie of substitution paths.

For fast retrieval, in the leaves of the substitution tree, MWS stores FormulaIDs. These are numbers uniquely associated with formulae, and they are also used to store context and occurences about the respective formula. They are stored in a separate LevelDB [7] database.

A simplified sketch of the index is shown in Figure 3.

MathWebSearch exposes a RESTful HTTP API which accepts XML queries. A valid query must obey the Content MathML format, potentially augmented with *qvar* variables which match any subterms. A *qvar* variable acts as a wildcard in a query, with the restriction that if two *qvar*s have the same name, they must be substituted in the same way.

TeMaSearch is using both MathWebSearch and Elasticsearch to answer queries. In order to achieve cooperation between the two systems, annotated documents are used. These annotated documents contain metadata from the original document (e.g. URI, title, author, etc.) and a list of FormulaIDs that can be found in that document.

2.3 Elasticsearch

Elasticsearch [3] is a powerful and efficient full text search and analytics engine, built on top of Lucene. It can scale massively, because it partitions data in shards and is also fault tolerant, because it replicates data. It indexes schema-free JSON documents and the search engine exposes a RESTful web interface. The query is also structured as JSON and supports a multitude of features via its domain specific language: nested queries, filters, ranking, scoring, searching using wildcards/ranges and faceted search.

The faceted search feature[2] is of particular interest to us. One way to use this feature is the terms aggregation: a multi-bucket aggregation, with dynamically built buckets. We can specifiy an array field from a document and ask ES to count how many unique items from the array are there in the whole index. This list can also be sorted, e.g. most frequently occuring items first. Additionally, we can also impose a limit on the number of the buckets (items) for which we want to receive the count.

An ES query which would return the most frequently used formulae (and subformulae) for “Pierre Fermat”, is presented in Listing 1. The key part is the *aggs* fields. We are specifying that we want an aggregation called *Formulae* on “terms” (i.e. we want bucket counting) and the target of the aggregation is the fields *ids*.

```
{
  "query" : {
      "match" : {
          "body" : {
              "query" : "Pierre Fermat",
              "operator" : "and"
          }
      }
  },
```

[2] Faceted search as such is now deprecated in ES and was replaced by the more powerful “aggregations”.

```
  "aggs" : {
      "formulae" : {
          "terms" : { "field" : "ids" }
      }
  }
}
```

Listing 1: Elastic Search Term Aggregation Query

A possible response to the above query can be found in Listing 2. In the response we can see the returned aggregations. In our example there is only one and it is called *formulae*. We can find the actual result in the *buckets* field. The key field in the bucket corresponds to a FormulaID. Here, the most frequent formulae were the one with ID 230 and the one with ID 93. The former appeared in 10 documents and the latter appeared in 9 documents.

```
{
    ...
    "aggregations" : {
        "formulae" : {
            "buckets" : [
                {
                    "key" : "230",
                    "doc_count" : 10
                },
                {
                    "key" : "93",
                    "doc_count" : 9
                },
                ...
            ]
        }
    }
}
```

Listing 2: Elastic Search Term Aggregation Response

2.4 Zentralblatt Math

zbMATH stores over 3 million entries corresponding to reviews or abstracts of mathematical documents, dating back to 1826 and drawn from more than 3,000 journals and 170,000 books[12].

Among the services it provides, there is the “structured search”. This is essentially a faceted search engine on the textual level. If we succeed in implementing it, we would like to integrate the Formula Schemata Generation Service with the zbMATH faceted search.

After running the current zbMATH faceted search implementation, a set of URIs (referencing scientific articles which answer the user’s query) will be provided. We should answer back with a set of formula schemata to be displayed to the user as another facet of the result.

2.5 arXiv

arXiv is a repository of over one million publicly accessible scientific papers in STEM fields. For the NTCIR-11 challenge [5], MWS indexed over 8.3 million paragraphs (totaling 176 GB) from arXiv. We will base our queries on this large index, because it provides a rich database of highly relevant formulae. Moreover, Elasticsearch will have more formulae on which it can run aggregations, also leading to more relevant results.

3 Implementation

This research project aims to develop a viable service for formulae schemata generation and to integrate it with existing services, ultimately providing formula faceted search. To the best of our knowledge, this problem has not been addressed before. In this section, I describe a possible solution for math faceted search.

3.1 Formalizing the problem

Let us now formulate the problem at hand more carefully.

Definition 1 *Given a set $\mathcal{D}$ of documents (fragments) – e.g. generated by a search query, a* ***coverage*** $0 < r \leq 1$*, and a* ***width*** n*, the* ***Formula Schemata Generation*** *(FSG) problem requires generating a set $\mathcal{F}$ of at most n formula schemata (content MathML expressions with* `qvar` *elements for query variables), such that $\mathcal{F}$ covers $\mathcal{D}$.*

Definition 2 *We say that a set $\mathcal{F}$ of formula schemata* ***covers*** *a set $\mathcal{D}$ of document fragments, with* ***coverage*** *r, iff at least $r \cdot |\mathcal{D}|$ formulae from $\mathcal{D}$ are an instance $\sigma(f)$ of some $f \in \mathcal{F}$ for a substitution σ.*

3.2 An Algorithm for FSG

The FSG algorithm we implemented requires a MWS index of the corpus. Given such an index, and a set $\mathcal{D}$ of formulae (as CMML expressions), we can find the set $\mathcal{F}$ in the following way:

- Parse the given CMML expressions similarly to MWS queries, to obtain their encoded DFS representations.
- Choose a reasonable cutoff heuristic, see 3.3.
- Unify each expression with the index, up to a given threshold (given by the above heuristic).
- Keep a counter for every index path associated with the unifications. Since we only match up to a threshold, some formulae will be associated with the same path (excluding the leaves). We increase the counter each time we find a path already associated with a counter.
- We sort these path-counter pairs by counter in descending order and take the first n (n being the width required by the FSG).
- If the threshold depth was smaller than a formula's expression depth, the path associated with it will have missing components. We replace the missing components with qvars to generate the schema and return the result set.

Figure 3 shows a simplified MWS index at depth 1. The formulae's paths represent their DFS traversal. Every formula can be reconstructed given its path in the index. The circles represent index nodes and the number inside represents the token's ID. When we reach a leaf node, we completely described a formula. This is encoded in the leaf node by an ID, which can be used to retrieve the formula from the database. The length of the arrows

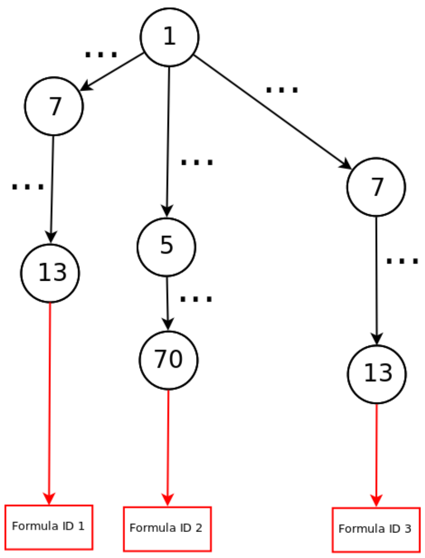

Figure 3: Simplified index at depth 1

symbolizes the depth of the omitted subterms (for higher depths, we have longer arrows). Notice how both formula with ID 1 and formula with ID 3 show the same "path" when ignoring subterms below a cutoff depth.

3.3 Finding a cutoff heuristic

In order to generate formula schemata, we must define a "cutoff heuristic", which tells the program when two formulae belong to the same schemata class. If there was no heuristic, two formulae would belong to the same class, only if they were identical. However, we want formulae that have something in common to be grouped together, even if they are not perfectly identical.

One reasonable cutoff heuristic would be a certain expression depth, given as a parameter to the schema-engine. In this way, a depth of 0 would always return $?x$ which corresponds to the 0-unification. The algorithm above is based on this heuristic. Another possible choice for the cutoff would be expression length. This would output formulae which begin the same way.

3.4 Integration with existing services

The core engine will accept a set of CMML expressions and return a number of schemata covering them. The two applications that follow immediately are Zentralblatt and Elasticsearch integration. Using the features described above, they are both capable of providing a set of hits belonging to some textual context. This would represent the faceted search at the textual level. From this set of hits, we need to obtain a set of CMML expressions, to feed into our core algorithm. The mechanism used to achieve this is described in the next section.

4 Evaluation

5 Conclusion

6 Applications and future work

References

[1] *ArXiv Online*. Dec. 21, 2014. url: http://arxiv.org/ (visited on 12/21/2014).

[2] *Bing Website*. Dec. 21, 2014. url: http://bing.com/ (visited on 12/21/2014).

[3] *Elastic Search*. Dec. 7, 2014. url: http://www.elasticsearch.org/ (visited on 12/07/2014).

[4] *Google Website*. Dec. 21, 2014. url: http://google.com/ (visited on 12/21/2014).

[5] Radu Hambasan, Michael Kohlhase, and Corneliu Prodescu. “MathWebSearch at NTCIR-11”. In: pp. 114–119. url: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/05-NTCIR11-MATH-HambasanR.pdf.

[6] Michael Kohlhase et al. “Zentralblatt Column: Mathematical Formula Search”. In: *EMS Newsletter* (Sept. 2013), pp. 56–57. url: http://www.ems-ph.org/journals/newsletter/pdf/2013-09-89.pdf.

[7] *LevelDB*. Dec. 21, 2014. url: http://leveldb.org/ (visited on 12/21/2014).

[8] *Mathematics Subject Classification (MSC) SKOS*. 2012. url: http://msc2010.org/resources/MSC/2010/info/ (visited on 08/31/2012).

[9] *MwsHarvest*. Dec. 21, 2014. url: https://trac.mathweb.org/MWS/wiki/MwsHarvest (visited on 12/21/2014).

[10] *Node.js*. Dec. 21, 2014. url: http://nodejs.org/ (visited on 12/21/2014).

[11] *Zentralblatt Math Website*. Dec. 7, 2014. url: http://zbmath.org/ (visited on 12/07/2014).

[12] *Zentralblatt Math Website*. Dec. 21, 2014. url: http://zbmath.org/about (visited on 12/21/2014).