Faceted Search for Mathematics

Radu Hambasan and Michael Kohlhase

Jacobs University Bremen

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.
In this paper, we describe one way of solving the faceted search problem in math: by extracting recognizable formula schemata from a given set of formulae and using these schemata to divide the initial set into formula classes. Also, we provide a direct application by integrating this solution with existing services.

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 or DuckDuckGo [3]) seem to be sufficient for the average user, they are limited when it comes to finding scientific content. The limitation arises because STEM[1] documents are also relevant for the mathematical formulae they contain and math cannot be properly indexed by a textual search engine. Math comprises of tokens that are expressed as structural markup (fractions, square-roots, subscripts and superscripts), which are not captured by simply indexing the text content of a page.

A good math search engine is therefore needed in several applications. For example, a large airline manufacturer 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 in the fields of physics and mathematics. 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,

 In: R. Bergmann, S. Görg, G. Müller (Eds.): Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB. Trier, Germany, 7.-9. October 2015, published at http://ceur-ws.org

[1] Science, Technology, Engineering and Mathematics

research and overall study process would be facilitated once they are provided with more than textual search. For all these applications, we first need 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]. Given such a high number of documents, with several million 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 of these is the Zentralblatt Math service [14].

Zentralblatt Math now employs formula search for access to mathematical reviews [7]. 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 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[2], 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 possibility of faceted search on the math content of the documents.

There are multiple ways in which we could understand a "math facet". One way would be through the MSC classification [10]. However, this would be rather vague because it will only provide information 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.

We are attempting to solve 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 user to drill in on *i*) variation theory and minimal surfaces, *ii*) higher-order unification, and *iii*) type theory. Following the MWS (see 2.1) tradition, the red identifiers stand for query variables, their presence making the results **formula schemata**.

These formula schemata were manually created to judge the feasibility of using schemata as recognizable user interface entities, but for an application we need to generate them automatically from the query. Moreover, each schema should further expand to show the formula class it represents. Formula classes would consist of all formulae sharing the same schema. This is the algorithmic problem we explore in this paper.

Acknowledgements This work has been supported by the Leibniz Association under grant SAW-2012-FIZ_KA-2 (Project MathSearch). The authors gratefully

[2] Mathematics Subject Classification

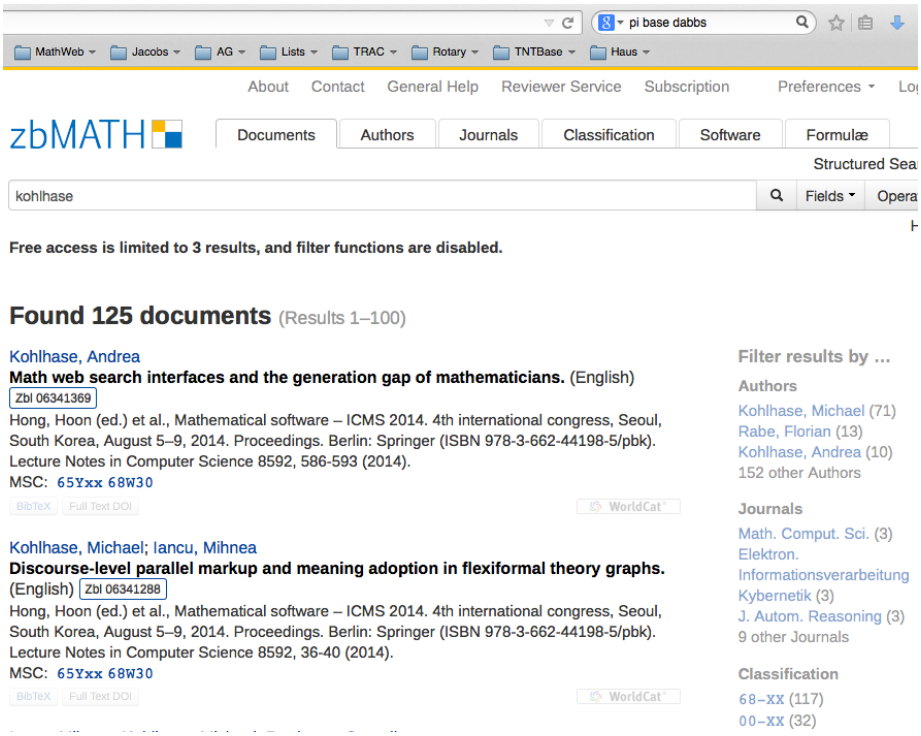


Fig. 1: Faceted Search in ZBMath

$$\int_M \Phi(d_p f) dvol$$
$$\lambda X.h(H^1 X) \cdots H^n X$$
$$\frac{\Gamma \vdash A \gg \alpha}{D}$$

Fig. 2: formula facets

acknowledge fruitful discussions with Fabian Müller, Wolfram Sperber, and Olaf Teschke in the MathSearch Project, which led to this research (the ZBMath information service uses faceted search on the non-formula dimensions very successfully) and clarified the requirements from an application point of view.

2 Preliminaries

In this section we describe the existent systems on which our work will be based, with the intention of making this paper self-contained.

2.1 MathWebSearch

At its core, the MathWebSearch [12] system (MWS) is a content-based search engine for mathematical formulae. It indexes MathML [9] formulae, using a tech-

nique derived from automated theorem proving: Substitution Tree Indexing [6]. Recently, it was augmented with full-text search capabilities, combining keyword queries with unification-based formula search. The engine serving text queries is Elasticsearch 2.2. 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 (Text + Math Search).

Internal to MWS, 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.

To facilitate fast retrieval, MWS stores FormulaIDs in the leaves of the substitution tree. These are integers uniquely associated with formulae, and they are used to store the context in which the respective expressions occurred. These identifiers are stored in a separate LevelDB [8] database.

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.

2.2 Elasticsearch

Elasticsearch [4] 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.

2.3 LaTeXML

An overwhelming majority of the digital scientific content is written using LaTeX or TeX, due to its usability and popularity among STEM researchers. However, formulae in these formats are not good candidates for searching because they do not display the mathematical structure of the underlying idea. For this purpose, conversion engines have been developed to convert LaTeX expressions to more organized formats such as MathML.

An open source example of such a conversion engine is LaTeXML [11]. The MathWebSearch project relies heavily on it, to convert arXiv documents from LaTeX to XHTML which is later indexed by MWS. It exposes a powerful API, accepting custom definition files which relate TeX elements to corresponding XML fragments that should be generated. For the scope of this project, we are

more interested in another feature of LaTeXML: cross-referencing between Presentation MathML and Content MathML. While converting TeX entities to Presentation MathML trees, LaTeXML assigns each PMML element a unique identifier which is later referenced from the corresponding Content MathML element. In this manner, we can modify the Content MathML tree and reflect the changes in the Presentation MathML tree which can be displayed to the user.

3 Schematization of Formula Sets & Implementation

In this section, we provide a theoretical description of the problem of generating formula schemata and a practical implementation.

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 $\mathcal{D}$ with coverage r.*

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 Defining a Cutoff Heuristic

To generate formula schemata, we must define a "cutoff heuristic", which tells the program when two formulae belong to the same schema class. If there is 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.

We experimented with several possibilities for the heuristic and found out that a dynamic cutoff which preserves the operators is optimal. We can identify the operators by looking at the first child of the apply token in the CMML tree. The user is given the option to have an absolute (fixed) or relative (depending on the depth of the CMML tree) cutoff for the operands.

Figure 3 illustrates this heuristic at depth 1. The divide element was kept, because it was the first child of apply, while the other children were removed. If we were to use a depth of 2, the plus element would also be included in the schema.

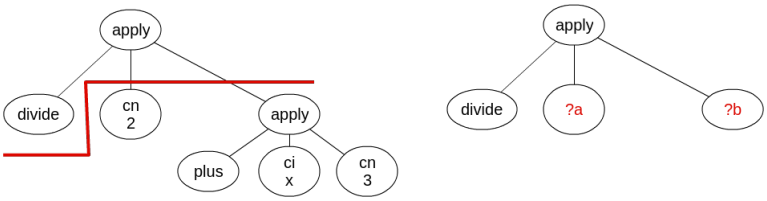


Fig. 3: Dynamic Cutoff

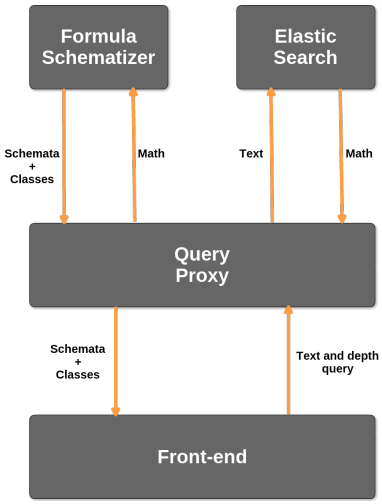


Fig. 4: FS Engine Architecture

3.3 Design Overview

The full faceted search system comprises of the following components: the Formula Schematizer 3.4, Elasticsearch, a proxy to mediate communication between the Schematizer and Elasticsearch and a Web front-end. The architecture of the system is shown in Figure 4.

Once the user enters a query (which consists of keywords and a depth), the front-end forwards the request to a back-end proxy. The proxy sends the text component of the query to Elasticsearch and receives back math contained in matching documents. Afterwards, it sends the retrieved math and the depth parameter (from the original query) to the Schematizer. The Schematizer will respond with a classification of the math in formula classes, as well as the corresponding schema for each class. Finally, the proxy forwards the result to the front-end which displays it to the user.

3.4 The Formula Schematizer

The Schematizer is the core part of our system. It receives a set of formulae in their Content MathML representation, generates corresponding formula schemata

and classifies the formulae according to the generated schemata. It provides an HTTP endpoint and is therefore self-contained, i.e. it can be queried independently, not only as part of the faceted search system. As a consequence, the Schematizer displays a high degree of versatility, and can be integrated seamlessly with other applications.

The central idea behind the schematization process is to generate signatures from formulae which can be used to identify formula classes. We use the MathWebSearch encoding for MathML nodes, where each node is assigned an integer ID based on its tag and text content. If the node is not a leaf, then only the tag is considered. The signature will be a vector of integer IDs, corresponding to the pre-order traversal of the Content MathML tree.

Naturally, the signature depends on the depth chosen for the cutoff heuristic. At depth 0, the signature consists only of the root token of the Content MathML expression. At full depth (the maximum depth of the expression), the signature is the same as the depth-first traversal of the Content MathML tree.

Based on these computed signatures, we divide the input set of formulae into formula classes, i.e. all formulae with the same signature belong to the same class. For this operation we keep an in-memory hash table, where the keys are given by the signatures and the values are sets of formulae which have the signature key. After filling the hash table, we sort it according to the number of formulae in a given class, since the signatures which cover the most formulae should come at the beginning of the reported result.

The Schematizer caller can place an optional limit on the maximum number of schemata to be returned. If such a limit was specified, we apply it to our sorted list of signatures and take only the top ones.

As a last step, we need to construct Content MathML trees from the signatures, to be able to show the schemata as formulae to the user. We are able to do this because we know the arity of each token and the depth used for cutoff. The tree obtained after the reconstruction might be incomplete, so we insert query variables in place of missing subtrees. We finally return these Content MathML trees with query variables (the formula schemata), together with the formulae which they cover.

3.5 The Front-End

To show the capabilities of the Schematizer we have prepared two demos. The first one is a text-only search engine which returns the math from the matching documents, after running it through the Schematizer. This is the demo for showcasing the schematization process. The second one is a direct application of the Schematizer into a Math Search Engine which is capable of mathematical faceted search.

SchemaSearch
The SchemaSearch front-end provides just a textual search input field. It is intended for users who want an overview of the formulae contained in a corpus.

The user can enter a set of keywords for the query, as well as a schema depth, which defaults to 3. The maximum result size is not accessible to the user, to prevent abuses and reduce server load. There is also an "R" checkbox which specifies if the cutoff depth should be absolute or relative. If relative, the depth should be given in percentages.

TemaV2
The TemaV2 front-end extends TeMaSearch to be able to perform mathematical faceted search. It is intended for users who want to filter query results based on a given facet (formula schema in this case). The look and feel is similar to the previous version of TeMaSearch, where the first input field is used to specify keywords and the second one is used to specify LaTeX-style formulae for the query. When returning results, a "Math Facets" menu will be presented to the user. We discuss this in Section 4.2.

3.6 Presentation by Replacement

After obtaining the schemata and formula classes, we need to be able to display the result to the user. One possibility would be to have the Schematizer return Content MathML expressions for the schemata and use an XSL stylesheet [13] to convert them to Presentation MathML. This approach would unfortunately generate unrecognizable schemata due to the inherent ambiguity of CMML. For instance, a csymbol element can be represented in several different ways depending on the notation being used. Additionally, we cannot reliably foresee all possible rules that should be implemented in the stylesheet and as a consequence some formulae will be wrongly converted.

Since the XSL conversion is unreliable, we will make use of the cross reference system provided by LaTeXML, as discussed in Section 2.3. Instead of returning Content MathML expressions, the Schematizer will use the first formula in each class as a template and "punch holes into it", effectively returning the ID of the nodes that are to be substituted with query variables. We will use this IDs to replace the referenced PMML nodes with `<mi>` nodes representing the qvars.

Figure 5 shows the presentation by replacement technique for a given schema. The Schematizer returned a schema which was checked against the first formula in its class ($\frac{2}{x+3}$) to generate two substitutions, marked with red on the left side. Due to the cross-reference system provided by LaTeXML, we are able to find the corresponding PMML elements and substitute them with `<mi>` tokens. The result will be displayed to the user as $\frac{?x}{?y}$.

4 Evaluation

4.1 SchemaSearch Front-end

Figure 6a shows the formula schemata at depth 3, over the arXiv corpus, for a query containing the keyword "Kohlhase". By default, the top 40 schemata are

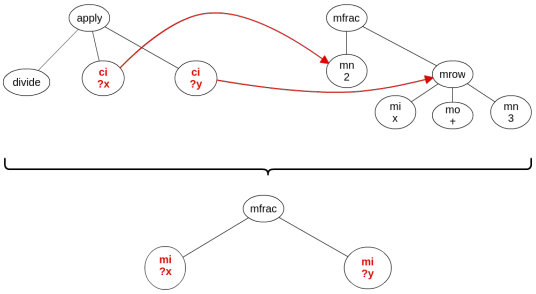
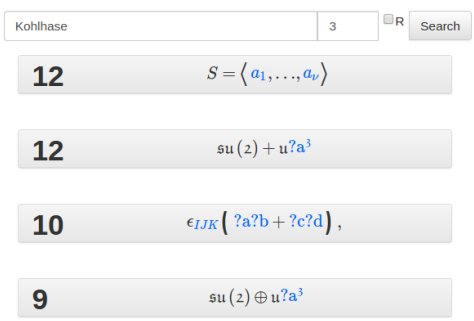


Fig. 5: Presentation by Replacement



(a) Faceted Results at depth 3

(b) Expansion of a Formula Class

shown, but the results are truncated for brevity. The bold number on the left side of each result item indicates how many formulae are present in each formula class. For instance, the third schema represents a formula class containing 10 formulae. The entities marked in blue are query variables (qvars).

Figure 6b shows the expansion of a formula class. There are 22 formulae in the class given by this particular math schema, as indicated by the count on the left upper side, out of which only ten are shown to the user (for brevity the class is truncated to 5 formulae).

We can see 2 unnamed query variables marked with blue as $?a$ and $?b$. By seeing the schema, the user can form an impression about the general structure of the formulae from that class. After expanding the class, the listing of concrete formulae appears. If the user clicks on one of them, he is redirected to the source document from which that expression was extracted.

4.2 TemaV2 Front-end

Figure 7 shows the results of a query for "Fermat" and $?a^{?n} + ?b^{?n} = ?c^{?n}$. Besides the regular TeMaSearch results, the user is also presented with a "Math Facets" section.

When the "Math Facets" section is expanded the user can see the top 10 schemata (ranked with respect to their coverage), as shown in Figure 8 (results

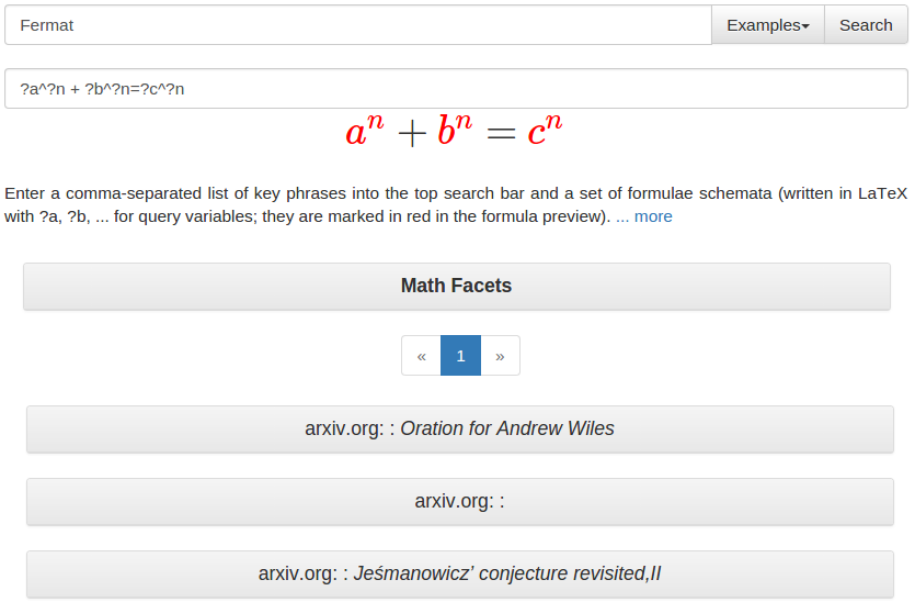


Fig. 7: TeMa v2 Query Results

truncated for brevity). We have also implemented a "search-on-click" functionality that allows the user the do a fresh search using the clicked schema and the initial keyword, which effectively filters the current results.

4.3 Performance of the Schematizer

We designed the Schematizer to be a very lightweight daemon, both as memory requirements and as CPU usage. To test if we achieved this goal, we benchmarked it on a server running Linux 3.2.0, with 10 cores (Intel Xeon CPU E5-2650 2.00GHz) and 80 GB of RAM.

We obtained the 1123 expressions to be schematized by querying Elasticsearch with the keyword "Fermat". While the overall time taken by the faceted search engine was around 5 seconds, less than a second was spent in the Schematizer. Also, the CPU utilized by the Schematizer never rose higher than 15% (as indicated by the `top` utility). Asymptotically, the algorithm would run in $O(N)$ time, where N is the number of input formulae. We are able to reach linear time performance, because each formula is processed exactly once and the signature is stored in a hash table, as discussed in Section 3.4.

Due to its implementation, the Schematizer is indefinitely scalable, because it does not require shared state between formulae and can therefore be implemented as a MapReduce [2] job, where mappers compute the signature of assigned formulae and reducers assemble the signature hash table.

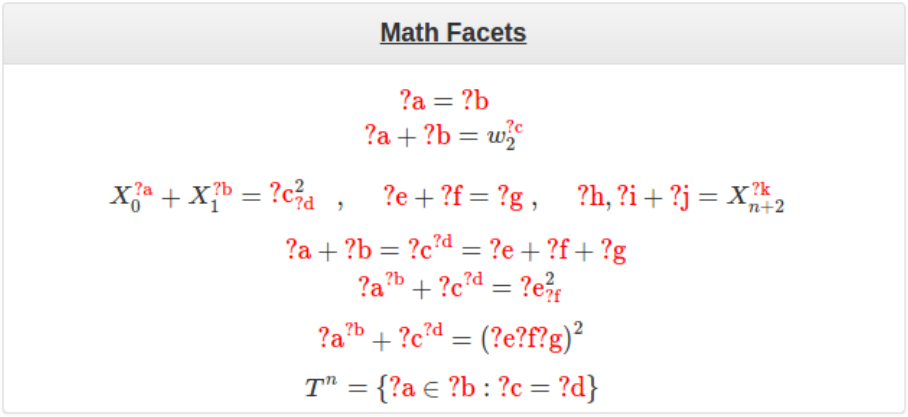


Fig. 8: Math Facets in TeMa v2

5 Future Work

One application of the faceted search engine can be providing mathematical definitions with the help of NNexus [5]. NNexus is an auto-linker for mathematical concepts from several encyclopedias, e.g. PlanetMath, Wikipedia. Assuming we are able to generate relevant schemata in response to keyword queries, we can target the faceted search engine with all the concepts stored by NNexus and store a schema for each such concept. Afterwards, for a given query, we can obtain the schema and check it against our stored set of schemata. If we find it, we can link the given expression to its mathematical definition. Given a large number of stored concepts and a high schemata relevance, the user should be able to see the definition of any encountered formulae on the Web. For example, hovering over $a^2 + b^2 = c^2$ will show the definition of the Pythagorean theorem.

Another, more direct, application of the Schematizer would be *Similarity Search*. One could create a MathWebSearch based search engine, which accepts an input formula and a similarity degree (between 0% and 100%). The engine would then create a formula schema at a relative depth corresponding to the similarity degree and use this schema to search the corpus. This approach defines the similarity between two formulae as the percentage of the CMML tree depth that they share.

6 Conclusion

We have presented the design and implementation of a system capable of mathematical faceted search. Moreover, we have described a general purpose scalable Schematizer which can generate intuitive and recognizable formula schemata and divide expressions into formula classes according to said schemata.

Although the Schematizer provides recognizable formulae, some queries to SchemaSearch (e.g. using an author as keyword) provide hits with a very low

relevance. This is because we cannot distinguish between the work of the author and work where the author is cited at the textual level. As a consequence, searching for "Fermat" would also show formulae from papers where Fermat was cited and if these papers are numerous, as it happens with known authors, would provide the user with misleading results. This suggests that a better source of mathematical expressions might be required for the SchemaSearch demo.

References

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

[2] Jeffrey Dean and Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. 2004.

[3] *DuckDuckGo Website*. May 8, 2015. URL: https://duckduckgo.com (visited on 05/08/2015).

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

[5] Deyan Ginev and Joseph Corneli. "NNexus Reloaded". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (Coimbra, Portugal, July 7–11, 2014). Ed. by Stephan Watt et al. Lecture Notes in Computer Science. Springer, 2014, pp. 423–426. URL: http://arxiv.org/abs/1404.6548.

[6] Peter Graf. *Substitution Tree Indexing*. 1994.

[7] 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.

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

[9] *Mathematical Markup Language*. URL: http://www.w3.org/TR/MathML3/.

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

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

[12] Corneliu C. Prodescu and Michael Kohlhase. "MathWebSearch 0.5 - Open Formula Search Engine". In: *Wissens- und Erfahrungsmanagement LWA (Lernen, Wissensentdeckung und Adaptivität) Conference Proceedings*. Sept. 2011. URL: https://svn.mathweb.org/repos/mws/doc/2011/newmws/main.pdf.

[13] *XSLT for Presentation MathML in a Browser*. Dec. 20, 2000. URL: http://dpcarlisle.blogspot.de/2009/12/xslt-for-presentation-mathml-in-browser.html#uds-search-results (visited on 04/04/2015).

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