ISSN 1870-4069

An Approach to Information Retrieval from Scientific Documents

Divyanshu Bhardwaj[1], Partha Pakray[1], Alexander Gelbukh[2]

[1] NIT Mizoram,
Department of Computer Science and Engineering,
India

[2] Instituto Politecnico Nacional,
Centro de Investigación en Computación,
Mexico

{divbhardwaj42, parthapakray}@gmail.com
gelbukh@gelbukh.com

Abstract. Information retrieval (IR) is one the prominent fields o6f research in the present day. Despite this, mathematical information retrieval represents a callow niche that isn't delved into much. This paper describes the design and implementation of a Mathematical Search Engine using Apache Nutch and Apache Tomcat. The search engine designed is to be able to exhaustively search for mathematical equations and formula in scientific documents and research papers. Processed Wikipedia texts were used as inputs in order to probe the ability to search for specific mathematical entities among large volume of text distractors.

Keywords: Math search engine, retrieval of scientific documents, mathematical expressions, canonicalizer.

1 Introduction

Information retrieval (IR) is the activity of obtaining information resources relevant to an information need from a collection of information resources. In simple terms, it is the tracing and recovery of specific information from stored data. While the recognition and retrieval of textual information is a well versed discipline, retrieval of mathematical expressions is in its nascent stage.

Mathematical Information Retrieval is concerned with finding information in documents that include mathematics. Mathematical expressions are of utmost significance in scientific documents. They provide a means for the proper promulgation of scientific information. Despite the importance of math in technical documents, most search engines do not support users' access to mathematical formulae in target documents.

As such the need for a mathematical search engine capable of exhaustively searching documents for specific expressions and formulae is genuine.

In this paper, we delve into our attempt of performing information retrieval on scientific documents mainly looking to perform retrieval of mathematical expressions from Wikipedia articles. The challenge of working with mathematical entities is the ability of the expressions to assume different forms and yet imply the same thing. This makes absolute searching of mathematical expressions strenuous.

The rest of this paper is organised as follows, Section 2 describes the related works. Section 3 gives the dataset description. Section 4 gives the detailed description of the system architecture. Section 5 details the NTCIR Math Task. Section 6 provides an introduction to MathML. Section 7 lists out the experiments while section 8 discusses the results. Section 9 sums up the conclusion and lists out the future endeavours.

2 Related Works

Mathematical Information Retrieval has been an important point of research in the past few years. It has been a significant part of the annual NII Testbeds and Community for Information access Research (NTCIR) Conference[1]. In NTCIR-10, the NTCIR-Math Task was organized as two independent subtasks, first being Math Retrieval and the second Math Understanding. Larson et al. [9] linked a classic keyword content information retrieval method with bitmap indexing of math operators.

Schubotz et al. [17] applied a distributed data processing system that accesses data in a non-index format. This allowed for fast processing of batch queries. Kohlhase and Prodescu [6] implemented a web service that provides low-latency answers to unification queries over content MathML expressions. Topic et al. [19] employed an indexing scheme for mathematical expressions within an Apache Solr database.

This resulted is results being returned even if the variables were different or the structure was changed. Liska et al. [11] implemented a similarity search based on enhanced full text search. Hagino and Saito [4] used indices which would hold structure information of math expressions in order to build a partial match retrieval system for math formulae.

In NTCIR-11, the Math-2 Task required to develop a common workbench for mathematical formula search. Schubotz et al. [20] worked upon evaluation of Math Similarity factors based on the results from the pooling process. Gao et al. [2] employed a system aimed at searching for mathematical formulae based on both textual and spatial similarities. Pinto et al. [14] used a combination of a feature extracted sequence mechanism of the formulae and a sentence level representation of the text describing the formulae to model the collection.

Hambasan et al. [5] amended upon their previously built MathWebSearch with the effort directed on scalability, integration of keyword and formula search, and hit presentation issues. Kristianto et al. [7] apart from implementing an indexing scheme for mathematical expressions within an Apache Solr database proposed a reranking method making use of textual information and dependency graph. Ruzicka et al. [15] presented their second generation of scalable full text search engine Math Indexer and Searcher with an integrated canonicalizer.

[1] http://research.nii.ac.jp/ntcir

Table 1. Corpus statistics.

Corpus	Articles	Math Entities
ArXiv	105,120	∼60M
Wikipedia	319,689	592,443

Pattaniyil and Zanibbi [13] made use of two indices, viz. a Solr/Lucene based index for document text , and a MySQL index for math expressions. Lipani et al. [10] employed four indices, one for text, and three for mathematical formulas. NTCIR-12, concentrated upon the relevance assessments for formula search.

Gao et al. [3] developed a system which aimed to retrieve mathematical information based on keywords, and the structure and importance of formulae in a document. Also employed was a hybrid indexing and matching model in which both keyword and structure information of formulae were taken into consideration. Kristianto et al. [8] further improved their previously built indexing scheme for math expressions within an Apache Solr database with added features such as three levels of granularity for textual information, a method for extracting dependency relationships between math expressions, score normalization, cold-start weights, and unification.

Ruzicka et al. [16] revamped their previously implemented MIaS with the new features aimed primarily at further canonicalizing MathML input, structural unification of formulae for syntactic based similarity search, and query expansion. Davila et al. [1] made use of two indices, a Solr-based index for document text and a custom inverted index for math expressions.

Thanda et al. [18] implemented LDA and doc2vec's co- occurrence finding techniques, in addition to pattern and elastic search-based document ranking. Results from knowledge bases with different scoring mechanisms were merged using a nested Borda Count based technique.

3 Data

The corpus upon which the information retrieval was performed was the Wikipedia corpus of NTCIR-12 MathIR Task[2] . The corpus was divided into *math* articles containing <math>tags, and *text* articles that do not. We only considered the <math>tagged articles for our experiment as our main motive was to retrieve mathematical expressions from Wikipedia text. These consisted of 31,839 articles which was approximately 10% of the whole dataset. These articles in turn contained 580,068 formulae. The description of the data set of the NTCIR-12 MathIR task is given in table 1.

4 System Architecture

In order to implement the interface for the retrieval of mathematical equations and formula from documents, we set up a search engine which would be able to exhaustively search for them in the crawled and indexed database.

[2] http://ntcir-math.nii.ac.jp/introduction/

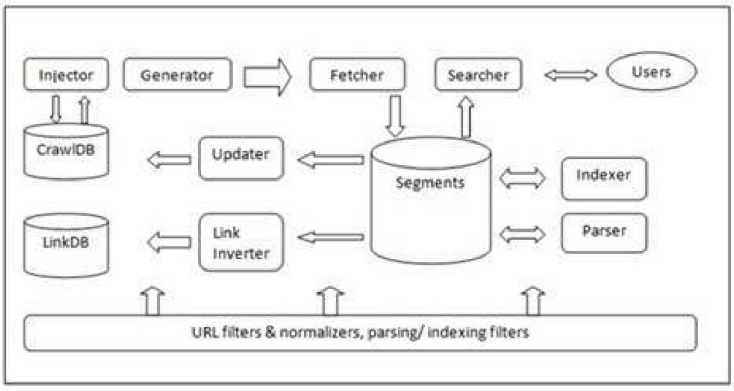

Fig. 1. System architecture of the Nutch based information retrieval system.

This search engine was entirely developed using Apache Nutch and run using Apache Tomcat. The graphical architecture of our Nutch implementation is given in Fig 1.

A prototype system was created by feeding the input data into Nutch. We implemented a generic Java code[3] to take the ids as the input to generate URL seeds. Next the injector injects the list of seed URLs into the *crawlDB*.

The generator then takes the list of seed URLs from *crawlDB*, forms fetch list and adds a crawl generate folder into the segments. These fetch lists are used by fetchers to fetch the raw content of the document. It is then stored in segments. Consequently the parser is called to parse the content of the document and parsed content is stored back in segments. The links are inverted in the link graph and stored in *LinkDB*.

This is followed by indexing of the terms present in segments and indices are updated in the segments. Following this information on the newly fetched documents is updated on the *crawlDB*. This crawl database is used as input in the implementation of the system. Queries are searched using this prototype system.

Results are obtained based on the ranking provided by Nutch. The architecture of Nutch as a web crawler can be interpreted from figure 2. The architecture of Nutch as a search engine can be interpreted from figure 3. We used the native ranking provided by Nutch for the development of the system. The ranking provided by Nutch may be explained using the following equation[4] :

$$\text{score}(\vec{q},\vec{d}) = \text{queryNorm}(\vec{q}) \times \text{coord}(\vec{q},\vec{d}) \times \text{norm}(t,\vec{d}) \times \sum_{t \epsilon \vec{d}} (tf(t) \times idf(t) \times t.\text{boost}(t.\text{field})), \quad (1)$$

where:

1. queryNorm() : indicates the normalization factor for the query,

[3] https://github.com/divyanshu42/IR/blob/master/ReadDirRecursevly.java
[4] Nutch And Lucene Framework- Arjun Atreya V RS-IITB

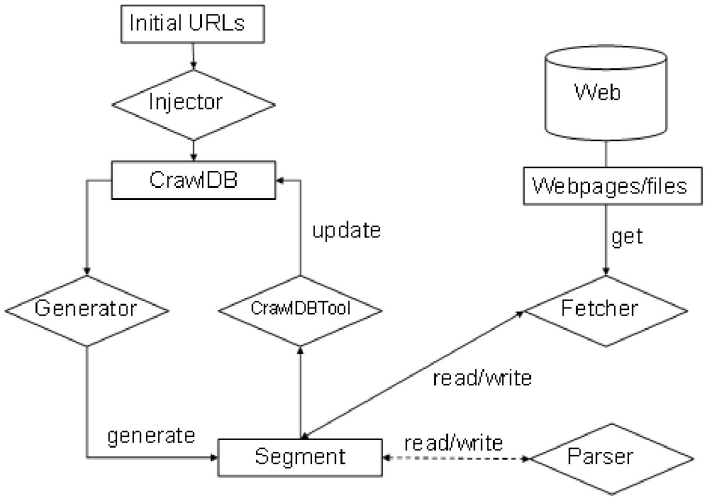

Fig. 2. Nutch as a web crawler.

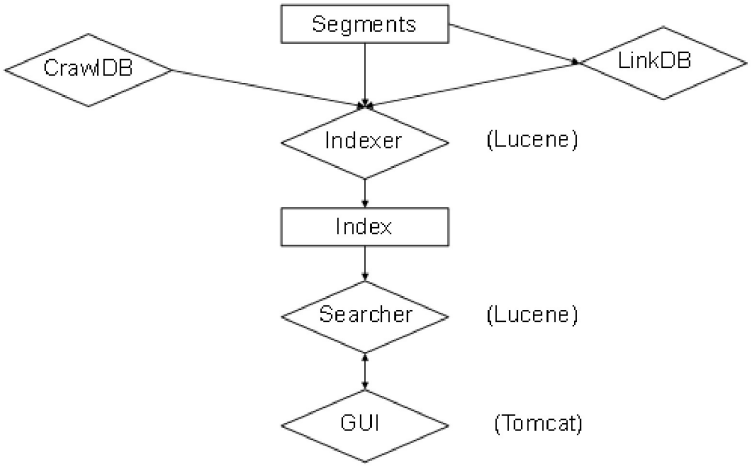

Fig. 3. Nutch as a complete search engine.

2. *coord() :* indicates how many query terms are present in the given document,
3. *norm() :* score indicating field based normalization factor,
4. *tf:* term frequency,
5. *idf:* inverse document frequency,
6. *t.boost() :* score indicating the importance of terms occurrence in a particular field.

5 NTCIR Math Task

NTCIR-12 Math Task is a shared task for retrieving mathematical information in documents in which queries are combinations of keywords and formula. Two corpora was used for the Math task. The first consisted of paragraphs from technical articles in the arXiv and the second consisted of complete articles from Wikipedia. For our system, we mainly utilised the Wikipedia corpus.

The arXiv dataset for NTCIR-12 MathIR consisted of scientific articles in English. Articles were converted from LaTeXto an HTML + MathML. The MathIR Wikipedia corpus contains articles from English Wikipedia converted into XHTML. 10% of the sampled articles contained explicit <math>tags that demarcated LaTeX. All articles with a <math>tag were included in the corpus.

The remaining 90% of the articles were sampled from Wikipedia articles that do not contain a *math* tag. These *text* articles functioned as distractors for keyword matching. Search topics were provided in a custom XML format. The topics consisted of the following items:
– Topic ID,
– Query (formula + keywords).

Given a set of queries, systems were to return a ranked list of search results for each of the arXiv and Wikipedia datasets.

6 MathML

Mathematical Markup Language (MathML)[5] is a mathematical markup language, an application of XML for describing mathematical notations and apprehending its structure and content. MathML is intended to reiterate mathematical and scientific content on the Web, and for other applications such as computer algebra systems, print typesetting, and voice synthesis.

MathML deals not only with the presentation but also the meaning of formula components (the latter part of MathML is known as "Content MathML"). Presentation MathML focuses on the display of an equation, and has about 30 elements. The elements' names all begin with m. A Presentation MathML expression is built up out of tokens that are combined using higher-level elements, which control their layout (there are also about 50 attributes, which mainly control fine details). Token elements generally only contain characters.

Content MathML focuses on the semantics, or meaning, of the expression rather than its layout. The <apply>element that represents function application. The function being applied is the first child element under <apply>, and its operands or parameters are the remaining child elements. Tokens such as identifiers and numbers are marked as *ci* and *cn*.

An example query from the NTCIR-12 math task in MathML is given in figure 4.

[5] https://www.w3.org/Math/

```
<topic>
 <num>NTCIR12-MathWiki-3</num>
 <query>
  <keyword id="w.0"> definition</keyword>
  <formula id="f.0">
   <math>
    <semantics>
     <mrow>
      <mi>a</mi>
      <mo>⊕</mo>
      <mi>b</mi>
     </mrow>
     <annotation-xml encoding="MathML-Content">
      <apply>
       <csymbol cd="latexml">direct-sum</csymbol>
       <ci>a</ci>
       <ci>b</ci>
      </apply>
     </annotation-xml>
     <annotation encoding="application/x-tex">a\oplus b</annotation>
    </semantics>
   </math>
  </formula>
 </query>
</topic>
```

Fig. 4. Example query in MathML to be retrieved by the system.

7 Experiment

We implemented a prototype system taking into account about 1GB of math tagged articles from the obtained data set. The steps involved in development of the prototype are mentioned in the underlying paragraph.

We start off by performing offline crawling for the data that is to be searched in. The crawl results in the formation of the crawl database known as *crawlDB.* The obtained crawl path is then added to the *nutch-default.xml* file in the *conf* directory.Next we build the nutch file using the *ant* and *ant war* commands.

Running these commands results in the the creation of the *war* file. This war file is then copied to apache tomcat folder *(-tomcat-(ver)/webapps/)*. We can now start our tomcat server. The apache tomcat server is started using the *catalina.sh start* command. In order to avoid the tedious task of running these commands each time, we created a shell script to automate it.[6] We now have a search engine up and running at the following address: *http://localhost:8080/nutch-(ver)/*

[6] https://github.com/divyanshu42/IR/blob/master/run.sh

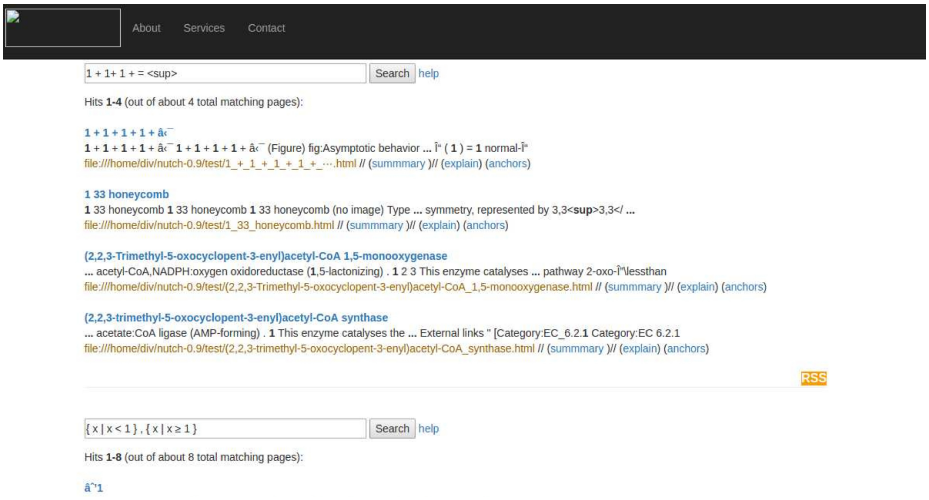

Fig. 5. Retrievals made by the system.

We implemented our search engine to search for a number of queries at the same time. This was achieved by modification of the search.jsp file which enabled us to supply a number of queries in a text file and search for them simultaneously.

During the experimentation, we tested the search engine for both mathematical expressions and text so as to compare performance of both the searches and also to be able to make sure that the search engine could be relied upon for textual searches.

8 Result

We obtained an extremely decent mathematical search engine which was capable of searching for mathematical expressions in the documents that comprised of the crawl database.

We were able to search for multiple queries in a single search which would prove to be extremely useful when searching for a number of math related equations and expressions at a given time. This has been shown in figure 5.

Upon analysis of the results we found that we obtained low precision in the retrieval for mathematical expression while we had relatively high accuracy for textual search.

For searching both math and text we implemented the same methodology, which only goes to draw attention to the challenges of working with mathematical expressions.

For instance a simple equation such as:

$$x^n + y^n = z^n. \tag{2}$$

Maybe represented in a number of different ways such as:

$$z^n = x^n + y^n, \tag{3}$$

$$or,$$

$$a^n + b^n = c^n. \tag{4}$$

Thus, the need of an efficient canonicalizer is genuine. The lack of such a canonicalizer meant that we could not retrieve all the different forms in which an expression may have been represented in some documents.

9 Conclusion and Future Works

In this paper we intended to delve into Information Retrieval (IR) in Scientific Documents by building a mathematical search engine which would be capable of searching exhaustively for mathematical expressions in scientific documents.

We made a lucid attempt at implementing an IR system and we obtained the expected results. While the results showed low precision for the mathematical expressions, this attempt proved to be a vital stride towards developing a fully functional and accurate retrieval system for mathematical entities in general. Now that we have a functional prototype capable of searching for mathematical expressions with decent accuracy and precision, our next endeavour would be working upon improving the precision of our system.

We plan to work upon a canonicalizer which would enhance the precision and give us better results while searching for mathematical entities. We would like to implement the canonicalizer and examine how it augments the precision and retrieval as a whole.

References

1. Davila, K., Zanibbi, R., Kane, A., Tompa, F. W.: The MCAT math retrieval system for NTCIR-10 math track. In: Proceedings of 12th NTCIR Conference (2016)
2. Gao, L., Wang, Y., Hao, L., Tang, Z.: ICST math retrieval system for NTCIR-11 math-2 task. In: Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-11, National Center of Sciences, pp. 9–12 (2014)
3. Gao, L., Yuan, K., Wang, Y., Jiang, Z., Tang, Z.: The math retrieval system of ICST for NTCIR-12 MathIR task. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, pp. 318–322 (2016)
4. Hagino, H., Saito, H.: Partial-match retrieval with structure-reflected indices at the NTCIR-10 math task. In: Proceedings of NTCIR Conference on Evaluation of Information Access Technologies (2013)
5. Hambasan, R., Kohlhase, M., Prodescu, C.: Mathwebsearch at NTCIR-11. In: Proceedings of 11th NTCIR Conference, pp. 114–119 (2014)
6. Kohlhase, M., Prodescu, C.: MathWebSearch at NTCIR-10. In: Proceedings of 10th NTCIR Conference, pp. 675–679 (2013)
7. Kristianto, G. Y., Topić, G., Aizawa, A.: The MCAT math retrieval system for NTCIR-11 math track. In: Proceedings of 11th NTCIR Conference, pp. 120–126 (2014)
8. Kristianto, G. Y., Topić, G., Aizawa, A.: MCAT math retrieval system for NTCIR-12 MathIR task. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, pp. 323–330 (2016)
9. Larson, R. R., Gey, F.: The abject failure of keyword IR for mathematics search: Berkeley at NTCIR-10 math. In: Proceedings of 10th NTCIR Conference, pp. 662–666 (2013)
10. Lipani, A., Andersson, L., Piroi, F., Lupu, M., Hanbury, A.: TUW-IMP at the NTCIR-11 math-2. In: Proceedings of 11th NTCIR Conference, pp. 143–146 (2014) doi: 10.13140/2.1.1127.8404

11. Liska, M., Sojka, P., Ruzicka, M.: Similarity search for mathematics: Masaryk university team at the NTCIR-10 math task. In: Proceedings of 10th NTCIR Conference, pp. 686–691 (2013)
12. Pakray, P., Sojka, P.: An architecture for scientific document retrieval using textual and math entailment modules. In: Proceedings of RASLAN 2014, pp. 107–117 (2014) doi: 10.13140/2.1.4036.2561
13. Pattaniyil, N., Zanibbi, R.: Combining TF-IDF text retrieval with an inverted index over symbol pairs in math expressions: The tangent math search engine at NTCIR 2014. In: Proceedings of 11th NTCIR Conference, pp. 135–142 (2014)
14. Pinto, J. M. G., Barthel, S., Balke, W. T.: QUALIBETA at the NTCIR-11 math 2 task: An attempt to query math collections. In: Proceedings of 11th NTCIR Conference, pp. 103–107 (2014)
15. Ruzicka, M., Sojka, P., Liska, M.: Math indexer and searcher under the hood: History and development of a winning strategy. In: Proceedings of 11th NTCIR Conference, pp. 127–134 (2014)
16. Ruzicka, M., Sojka, P., Liska, M.: Math indexer and searcher under the hood: Fine-tuning query expansion and unification strategies. In: Proceedings of 12th NTCIR Conference, pp. 331–337 (2016)
17. Schubotz, M., Leich, M., Markl, V.: Querying large collections of mathematical publications NTCIR10 math task. In: Proceedings of 10th NTCIR Conference, pp. 667–674 (2013)
18. Thanda, A., Agarwal, A., Singla, K., Prakash, A., Gupta, A.: A document retrieval system for math queries. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, pp. 346–353 (2016)
19. Topic, G., Kristianto, G. Y., Nghiem, M. Q., Aizawa, A.: The MCAT math retrieval system for NTCIR-10 math track. In: Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies (2013)
20. Youssef, A., Schubotz, M., Markl, V., Cohl, H. S., Li, J. J.: Evaluation of similarity-measure factors for formulae based on the NTCIR-11 math task. In: Proceedings of 11th NTCIR Conference, pp. 108–113 (2014)