Math Retrieval Using Leaf-Root Expression Trees

Andrew Norton*
University of Virginia
Charlottesville, Virginia
apn4za@virginia.edu

Ben Haines†
University of Virginia
Charlottesville, Virginia
bmh5wx@virginia.edu

ABSTRACT

This paper describes our implementation of a system for searching for mathematical expressions. The paper summarize existing techniques for math retrieval and describes our particular implementation. We propose a few extensions to implement and provide experimental results that measure the effectiveness of these proposals.

1. INTRODUCTION

1.1 Background

Within the field of Information Retrieval, the task of mathematical expression retreival is a topic that has been attracting an increasing amount of attention in recent years. Math search is useful in a variety of situations, particularly for students and practitioners of technical fields. Particular scenarios include researchers who want to discover relevant work relating to a particular function or a student who needs help solving a particular problem. Existing solutions are often unsatisfying. For example, consider arxiv.org and math.stackexchange.com. These sites are two of the largest resources of collected mathematical information for both professional researchers and student. However, the usefulness of much of this information is decreased by challenges in locating it. The search feature of the arXiv does not permit searching for commonplace symbols such as "+" or "-". Searches on stackexchange often return no relevant results despite additional efforts revealing that multiple relevant results do exist. A better math retrieval system could in both cases increase productivity of many users.

Multiple approaches to parsing, indexing, and searching mathematical expression have been proposed. A specific state of the art algorithm does not exist as research in the field has not had time to converge to optimal solutions for the problems of storing and parsing expressions. Approaches tend to fall into one of two categories, those that use established text based search methods with modifications in order to apply them to the particular problem of Mathematical Information Retrieval (MIR) and those that use tailored approaches that attempt to take advantage of the inherent structure of expressions to improve performance. We provide a brief comprison of techniques in these categories and justify our particular choice of a system to implement. Our research focuses on the effectiveness of the Leaf-Root path system for representing structured expressions. The primary contribution of the work is the suggestion of query expansion in order to allow searches for generalized expressions and an examination of how changes in the parsing grammar can effect performance.

2. RELATED WORK

Among the earliest discussions of a math retrieval system, and one of the few describing an actual large scale implementation is the paper by Youssef and Miller regarding the Digital Library of Mathematical Functions[3]. The idea proposed involves a sequence of steps to process mathematical notation into a format recognizable by existing search engines. This involves first using macros to map math symbols to standard alphanumeric text representations. For example "+" and "<" are mapped respectively to "plus" and "lt". Next, nested expressions, such as exponents, are flattened. Finally, expressions are normalized by sorting the leaves of the corresponding parse tree in a standardized manner.

The authors select an evolutionary approach that augments existing text search engines due to practicality constraints but they also outline a few relevant challenges that they suggest could be better addressed with a structural approach. These challenges include

- Recognition of mathematical symbols
- Capturing and indexing structure
- Accounting for mathematical "synonyms"

Since the 2003 publishing of the paper discussed above, incremental improvements have been suggested by a variety of sources. In 2007 Miner and Munavalli introduced a more involved for processing inputs while still relying on a standard text search infrastructure for the fundamental search operation.[4] Largely within the last five years a number of papers have emerged that attempt to address the problems in more fundamental ways. In 2012 an approach that bridges the gap between text and structure based methods is introduced in the paper "A structure based approach for mathe-

*
†

CHARLOTTESVILLE '15 Charlottesville, Virginia USA

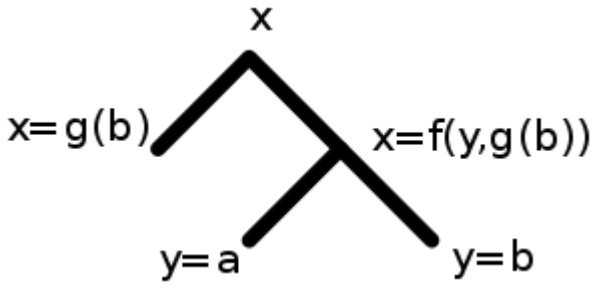

Figure 1: A sample substitution tree

matical expression retrieval".[5] The fundamental idea is to use a modified longest common substring (LCS) algorithm to measure similarity between expressions. The expressions are tokenized and each token is given an label to indicate its nested depth. The LCS algorithm then weights the similarities between two expressions by how closely depth markers are aligned. An alternate structure capturing approach that has been explored in a number of papers is that of substitution trees.[2][1] In such a tree each internal node represents a generalized expression form and leaves represent specific expressions for which all variables have a substituted value. A simple example tree is shown in the next figure. This structure stores relationships between abstract expression forms. Within each node individual expressions are stored as symbol layout trees that map spatial relationships between individual components of an expression. The authors of this approach reported performance improvements when compared to a standard Lucene search that they attribute to substructure queries enabled by the tree structures.

The approach that most directly influenced our implementation is an alternative method for using trees to store structure. Furthering our intuition that parse trees could be a useful representation of structure the particular implementation by Zhong provided inspiration and solutions to some technical challenges.[6]

Despite these recent innovations it is interesting to note that the top performers on MIR tasks, such as the main task of NTCIR11 are still acheived by traditional text based systems with modifications.

3. PROPOSED SOLUTION

There are a number of general components to our proposed solution. Each will be discussed separately and the particular decisions we made will be justified.

3.1 Representation

The system was inspired by and particularly targets the Stack Exchange and arXiv use cases which both store math information as LaTeX. Additionally, the corpus of expressions made available to us contains LaTeXexpressions. For these reasons we chose to focus on searching for LaTeXexressions rather than alternatives such as MathML. As a result of this we decided to accept LaTeXformatted inputs. The advantage of this choice is that LaTeXis a widely used system for expressing mathematical structures, in particular, users who want to search a LaTeXcorpus are probably also capable of composing queries with it. As well as this, many simple expressions can be given without any overt formatting and will still be valid which increases ease of use.

When given a raw string, whether it be as part of the corpus or in a query the system first performs a number of preprocessing operations. First, operations such as "\left and \right" purely change the display of an expression and not the structure or content are removed. Second, semantically similar operations are grouped roughly into equivalence classes and occurrences are replaced with a single symbol. For example, "\pmod, \bmod, and \mod" which have the same meaning but different presentations are replaced with the common command "MOD". Other operations that might have clear differences in meaning but are also similar in some way such as "+, \sum, and \bigoplus" are replaced with the common command "SUM". Numbers are all normalized to the symbol "NUM" and variables to "VAR".

After this preprocessing step the expression is passed through a parser designed to recognize latex constructs. We used Antlr to generate the parse from a language specification. A complete specification was included in the Cowpie project and we used a modified subset for simplicity.

3.2 Index Construction

After the expressions have been processed we are left with a collection of tree structures that need to be organized into some way to allow effective structure based search. This is the key component of the leaf-root tree approach. Each expression is given a unique ID number and its tree is decomposed into a collection of leaf root paths. For example consider the parse tree displayed in figure 23.2 corresponding to the expression "2 * (3 + 4)". After normalization the tree is decomposed into the two distinct paths "2→ TIMES" and "NUM→ PLUS→ TIMES". These paths provide a loose approximation of the structure of the tree as a whole. There are a few immediate advantages of using parse trees and in particular leaf-node paths. The first is that they assume commutativity of operations. If the expression in figure two was instead "2 * (4 + 3)" its representation as a collection of leaf-root paths would be unchanged. Although there are obvious exceptions this is a good general assumption as it is true for most common operators. The second and more important property is that it inherently supports searching for subexpressions as they form subtrees. Given two expressions E_1, E_2, if E_1 is a subexpression of E_2 then all of its paths will also be subpaths of $E2$'s paths. Any tree that contains a subexpression of the form NUM * (NUM + NUM) is guaranteed to have these paths as subpaths. It's important to note that the paths do not provide a complete description of the tree, some information is lost in their creation. That is, it is possible to have trees that represent distinct expressions but which decompose into the same paths. Thus the paths are used as a first filter to reduce search space rather than a final retrieval mechanism. In order to take advantage of the paths, we structure the index as a filesystem. In our implementation we used the computer's actual filesystem for convenience but this could be implemented in a ore efficient way for this dedicated task. Sub-directories are created for each node on the path, beginning at the leaves and the ID of the expression is stored at the deepest locations. For example, the sample expression would have its ID stored in ./NUM/TIMES and ./NUM/PLUS/TIMES.

The IDs themselves are used as keys to lookup information about the particular expression in an external structure. This might include the original typesetting, the actual

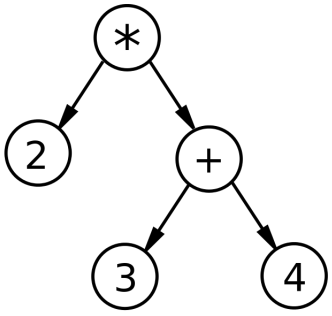

Figure 2: A sample expression tree

parse-tree, the source webpage, or other relevant information depending on the context that the search is operating in.

3.3 Queries and Retrieval

As explained in the representation section, queries are accepted in LaTeXformat. They are subjected to the same process of normalization and parsing as the corpus expressions are and decomposed into leaf-root paths. In order to retrieve relevant expressions from the corpus we just traverse the index according to each of paths. Continuing the example from above, assume that the index contains only the expression from figure 2 and that a user has queried for "5 + 2" which breaks down to the single path "NUM→ PLUS". We first traverse the filesystem to the location ./NUM/PLUS and then return the merged contents of all of the subdirectories. This process gives us every expression for which the query is a sub-part.

Depending on the query, and particularly for short queries, the returned set of expressions is a potentially significant portion of the entire corpus. Thus a method is needed for ranking within this filtered set. We use a few different approaches based on the structure of the trees in order to rank, and we also reintroduce the symbolic information removed in the first parsing step in order to compare the literal terms of the query.

DESCRIBE RANKING PROCESS IM NOT TOTALLY SURE HOW IT ACTUALLY WORKS.

4. EXPERIMENT

The lack of an annotated corpus and the difficulty involved in creating one put limitations on our ability to use standard metrics to evaluate the system's performance. As a results our testing procedure consisted of selecting a "target" expression and then searching for modified versions of the expression or manually created expressions that we deemed "relevant" and observing if the target expression is returned. Some examples are presented here and general comments are given.

1. PUT EXAMPLES HERE 2. PUT COMMENTS HERE 3. IF POSSIBLE PLEASE PUT SOME EXAMPLES OF QUERY EXPANSION ACTUALLY DOING SOMETHING USEFUL HERE. 4. JUSTIFICATION THAT WE DID SOMETHING OTHER THAN POORLY REIMPLEMENT WOULD BE V HELPFUL.

5. LIMITATIONS

There are a number of limitations associated with our work here. Some of them are inherent to the field, some are a result of our choice of approach, and some are of our particular implementation. Due to the newness of the field there is a scarcity of mature tools for solving this kind of problem. There are also few established or annotated datasets available. Due in particular to the problem of identifying mathematical synonyms, it is difficult to create relevance judgements for arbitrary queries.

1. BETTER RANKING FORMULA - preserve symbols, e.g. x + y = x + 3 <- x 2. Better index - don't use native fs 4. Only does symbols not symbols and words.

3. Our grammar/parsing is weak and limits functionality to a not very interesting subset of the data.

6. CONCLUSIONS

1. I DON'T KNOW WHAT OUR CONCLUSIONS ARE. CONCLUSIONS ARE PROJECTS SUCK EXAMS4LYFE.

2. GET THE BIBFILE WORKING I'M NOT SURE HOW IT IS SUPPOSED TO BE.

6.1 Citations

Citations to articles [**?**, **?**, **?**, **?**], conference proceedings [**?**] or books [**?**, **?**] listed in the Bibliography section of your article will occur throughout the text of your article. You should use BibTeX to automatically produce this bibliography; you simply need to insert one of several citation commands with a key of the item cited in the proper location in the `.tex` file [**?**]. The key is a short reference you invent to uniquely identify each work; in this sample document, the key is the first author's surname and a word from the title. This identifying key is included with each item in the `.bib` file for your article.

The details of the construction of the `.bib` file are beyond the scope of this sample document, but more information can be found in the *Author's Guide*, and exhaustive details in the *LaTeX User's Guide*[**?**].

This article shows only the plainest form of the citation command, using `\cite`. This is what is stipulated in the SIGS style specifications. No other citation format is endorsed or supported.

7. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

8. REFERENCES

[1] R. Z. Chair, B. Y. Reader, M. T. Schellenberg, M. T. Schellenberg, and R. Zanibbi. Layout-based substitution tree indexing and retrieval for mathematical expressions, 2011.

[2] M. Kohlhase and I. Sucan. A search engine for mathematical formulae. In J. Calmet, T. Ida, and D. Wang, editors, *Artificial Intelligence and Symbolic Computation*, volume 4120 of *Lecture Notes in*

Computer Science, pages 241–253. Springer Berlin Heidelberg, 2006.

[3] B. Miller and A. Youssef. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 38(1-3):121–136, 2003.

[4] R. Miner and R. Munavalli. An approach to mathematical search through query formulation and data normalization. In M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors, *Towards Mechanized Mathematical Assistants*, volume 4573 of *Lecture Notes in Computer Science*, pages 342–355. Springer Berlin Heidelberg, 2007.

[5] P. Pavan Kumar, A. Agarwal, and C. Bhagvati. A structure based approach for mathematical expression retrieval. In C. Sombattheera, N. Loi, R. Wankar, and T. Quan, editors, *Multi-disciplinary Trends in Artificial Intelligence*, volume 7694 of *Lecture Notes in Computer Science*, pages 23–34. Springer Berlin Heidelberg, 2012.

[6] W. Zhong. A novel similarity-search method for mathematical content in latex markup and its implementation, 2015.