

A Search System for Tree-Structured Mathematical Formula in *LaTeX* Format

Wei Zhong, Hui Fang
Dept. of Electrical and Computer Engineering
University of Delaware
Newark, DE USA
{zhongwei, hfang}@udel.edu

ABSTRACT

The special characteristic of mathematical language, compared with normal text content, makes mainstream retrieval models (e.g. “bag of words” model) deficient to provide good result in cases where query is in mathematical language. In this paper, we address the difficulties of similarity identification of mathematical contents and present a search system specifically for mathematical contents, more precisely, a system to tokenize mathematical content in *LaTeX* into formulas, to transform a tree-structured formula to what we call “branch words”, and to score the relevance degree between “branch words”. Using this system, we show the potential and possibility to tackle the problems in mathematical searching.

Categories and Subject Descriptors

H.3 [Information Search and Retrieval]: Miscellaneous

General Terms

Algorithms

Keywords

mathematical searching, language processing, search engine

1. INTRODUCTION

Unlike normal text content, mathematical language, by its nature, has many differences from normal text content. First, one mathematical formula may use different notations and thus has different representations.¹ Second, two semantically different mathematical document can contain the same number of terms.² Also, the order of terms in math language sometimes matters but can be mutable in other cases, which implies it is impractical to uniformly apply one single language model to mathematical content.

¹An example, $a^2 + b^2 = c^2$ can be described as $x^2 + y^2 = z^2$ as well.

²Consider $ax + (b + c)$ and $(a + b)x + c$.

Currently existing and working mathematical searching systems that we can find online, primarily are *WolframAlpha*³, *(uni)quation*⁴ and *MWS*⁵. *WolframAlpha* mainly focus on computation and evaluation of mathematical equation input rather than searching the similar equation. *(uni)quation* is able to solve the problems addressed above in this section but it is not under development and the model and method it uses is currently unclear.⁶ *MWS* uses Term indexing[2, 3] which we are not using here, to discriminate the mathematical structures.

Our system tries a different approach to make use of the structure of mathematical formula to solve the problems addressed above. It uses an efficient way to parse and tokenize mathematic formula, to transform a tree-structured formula to a more comparable structure, and also gives the method to score and rank results.

2. SYSTEM DESCRIPTION

Our system has a WEB front-end to accept user input in *LaTeX* format and pass it to the back-end. The back-end, mainly consists of a parser and a search program. The function of parser is to do tokenization and tree construction as well as storing output structure into our collection. The search program will then compare the query and document and evaluate the similarity of them, and give the final ranking through output file for the WEB front-end CGI program⁷ to read. We input mathematical content in *LaTeX* as document to our parser by either manually selecting or a crawler script specifically targeting at mathematical content website *Mathematics Stack Exchange*⁸.

2.1 Tokenization and Tree Construction

We choose to tokenize a subset of mathematic related *LaTeX* language, we use *Lex/Yacc* tools to tokenize the *LaTeX* language and construct a “tree” for each equation. The LALR parser generator of *Yacc* can handle language efficiently in guaranteed linear time[1]. The grammar we use will parse mathematical content into different classes of tokens including variables, different basic mathematical operators, equal class, times class, fraction class and square root. And we

³<http://www.wolframalpha.com/>

⁴<http://uniquestion.com/>

⁵<http://search.mathweb.org/>

⁶<http://rystsov.info/2009/05/01/uniquestion.html>

⁷Common Gateway Interface, a way for Apache web server to interact with external programs.

⁸<http://math.stackexchange.com/>

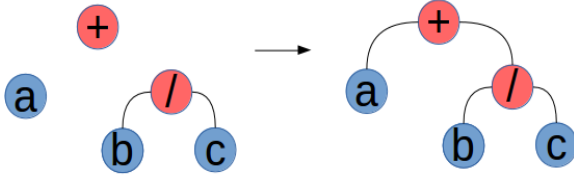


Figure 1: Example of Sub-tree generation for the addition grammar.

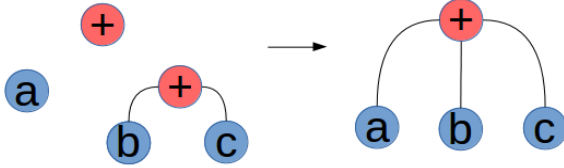


Figure 2: Cases where commutative property applies.

choose to omit undefined control sequence for the sake of robustness.

When a grammar is reduced, the tokens is converted to a tree node directly or by attaching sub-trees which is reduced previously to the new root. In this way, we will finally get a tree structured representation.

Some operations may have commutative property, in these cases⁹, all the sons in two adjacent levels will be attached to the same root.

2.2 Extraction of Branch words

After constructing a tree, A “branch word” is extracted by taking tokens from the leaves to the root of a tree in order. Branch words are used to be compared with those of other trees.

⁹In our system, the cases where we apply commutative property include *addition* and *multiplication* operations.

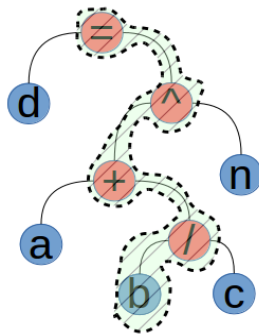


Figure 3: Illustration of One Branch Word for formula $(a + \frac{b}{c})^n = d$.

We notice that one math equation can use different symbol sets, so we choose not to distinguish the leaves’ actual symbol in the branch word. We also notice the branch word is not enough to distinguish trees, an example would be the equation $(a + b + c) \times (d + e)$ and $(a + b) \times (c + d + e)$, they have the same branch words yet have different semantic meaning in mathematical language. To avoid this flaw we further introduce the weight for any node n_i in a branch word, defined by the sum of that of its successors, which is given by:

$$w(n_i) = \sum_n 1, \quad n \in \{n_i \cup succ(n_i)\}$$

2.3 Indexing and Storage

The storage of mathematical formulas in our system, contains all the branch words from each tree as well as the weight information of each node in branch words. To enable efficient retrieval, branch words are stored in file system where the path is named by token names of a branch word (with weight information) in order¹⁰. The path where a branch word resides also store a “posting” file recording all the documents in the collection that contain that branch word.

2.4 Comparing and Scoring

In our system, comparing two pieces of mathematical content is essentially to compare all the branch words from one content with those of the other, calculate the similarity degree between two branch words. In terms of search system which rank-orders the documents matching a query, a score with respect to the query for each matching document is computed by sum the similarity degree for each related document in the document collection, then rank all the related documents using the sum score.

Here we give our formulas to both calculate the difference degree between two branch words and score each related document. For the detailed formula, we have to put some notations: Let m be the number of continuous matches between two branches from the beginning of the branch word, n be number of same branch word, l be the length of branch word. Then for query branch word i and document branch word j , we use:

$$\begin{cases} s_{i,j} &= \min(n_i, n_j) \times \frac{m}{l_i} + \frac{1}{|n_i - n_j| + 1} \times \frac{m}{\max(l_i, l_j)} \\ S &= \sum_{k \in T} s_{i,j}^k \end{cases}$$

3. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these

4. REFERENCES

- [1] D. E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6), 1965.
- [2] W. W. McCune. Experiments with discrimination-tree indexing and path indexing for term retrieval. *Journal of Automated Reasoning*, 9(2), 1992.

¹⁰One example can be `./collection/var/frac/add`

- [3] Stickel. The path-indexing method for indexing terms.
In *Technical Report*, page 473. Artificial Intelligence
Center, SRI International, October 1989.