Issues to solve in the equation search

April 8, 2011

1 Introduction

The task of the mathematic formula search impose several problems. Some of them include:

1. **Absence of clear definition of proximity between two formulas**. It is not obvious should the equations $c = a + b$ and $\ln(1 + \frac{b+a}{-c}) = -\infty$ (being equal from math point of view) be considered equal or not and should they be considered closer to each other than for instance the first formula to the equation $a = b + c$, which has the same parameters but different meaning, though looks much more similar to $c = a + b$ than the second equation.

2. **The search query**. If search is intended for usage by people it should be accessible and understandable to use. In text search simplest search query just contains words in the natural language – something that every literal person can do. For the formula search the query language is more complex task: for the ability to conduct any reasonable search a user should be able to use the widest possible range of math notation.

3. **Dependence on context of formula usage**. Sometimes it is impossible to determine the meaning of symbolic literals in a formula without seeing the context in which some functions, variables or constants may be defined. Trying to look only at the formula content itself may lead to both false-positive and false-negatives in an equality estimation.

4. **Relevance ranking in fuzzy search**. Standard conventional method which are being used for text search (like BM25) do not fit well enough to the task of formula search without heavy modification because they normally rely on the statistic of the documents and assume that the average document has far more tokens than the search query.

5. **Parsing formulas given in TEX syntax**. This problem is not general for the formula search task but more for the specific application. However it exists: TEX syntax for math expressions is indeed presentational. In most cases it is possible to reproduce the exact mathematical meaning of the formula but for some corner cases heuristics may be needed to fix mistakes in the formula definition (like typing `sin` instead of `\sin`) and for some formulas (which may still look normal in TEX) the exact math meaning can not be extracted or may even become distorted.

2 Different approaches

There are several possible ways to define the proximity measure between two formulas, some of them are reviewed in the following sections.

2.1 Feature extraction

The most straightforward way for defining the proximity measure is feature extraction (also called "template/pattern matching" in some articles, see eg. [1]). The main idea of it is in treating the formula as the container of some predefined patterns, such as "integral", "square root" or "plus operator". It is trivial to extract such patterns from formulas defined in LaTeX (or MathML as in [1]) using simple lexer and tokenizer or even with regular expressions (though it could never be ideal).

Each feature may then get a unique index (the dimension number). For each formula then we may calculate its *feature vector* as a list of coefficients each related to corresponding feature. Values in the vector's dimensions could be defined just as of boolean type (1 for feature being in formula and 0 for features which absent) or may represent some relative importance of a feature (for example the number of usages in the formula or even the feature's tf-idf in the representative formulas corpus).

In a result we have a n-dimensional vector space (where n is the number of predefined features) and a feature vector for each formula. For a search we may make the query syntax to be the same as the source formula syntax so the same routines could be made with a search query and its feature vector could be retrieved as well. The similarity between the formulas can be easily estimated as the cosine of the angle between two vectors which in practice is being calculated as a dot product of length-normalized vectors.

This approach is not specific to the formula search and is in fact a general way to fight the "curse of dimensionality". Though for the mathematic expressions this principle helps a lot in providing a clear way to rank search results and to calculate the distance between any arbitrary formulas, it still has some shortcomings in such specific application. First, all the features should be carefully analyzed and defined in advance. If a feature was not predicted by the time the algorithm was written it will never support it. Second, it is very limited in the semantic analysis of the formula content and in a simplest case does not count the order and nesting of the operators and functions. It is still possible to add to the templates list something like "sinus inside of the integral" or "square root of the sum of squares" but it would only increase the complexity and quantity of the patterns (and in the end the number of dimensions). And even then it would be very limited in terms of alternative forms of the same math concepts and any other high-level analysis which in any way requires the extraction of the semantic information from the formula, not just the the list of operators and functions it uses. Moreover, pure feature extraction method offers no way to add context sensitivity to the formula search. Also the approach seems to be prone to error in comparison between two complex equations because it will consider them close just because they have similar set of features.

2.2 Building and comparing semantic trees

More sophisticated approach in calculating the similarity between the formulas should be based on understanding of the semantic meaning of the formula and interpreting it as a math expression and not as a list of characters (see for example [2]).

To make it possible it is required to reproduce the original formula's semantics from possibly only limited presentation notation. During the formula parsing the tree which represents the equation content should be built. Its structure may be similar to Content MathML dialect, i.e. it may have node for each operator (or function) and the arguments of function (or operator) as its children nodes. At this point some simple symbolic computation can be performed, for example assigning the whole equation to 0 at the left side, permuting sum terms or replacing the square root with a power of 1/2.

The proximity calculation task boils down to the tree comparison either with a tree merging algorithm or with what [2] calls *substitution-tree indexing*. The first seems to be more flexible and capable to fuzzy search and subterm search while the second offers lower memory footprint and less query execution time.

Still this method itself does not imply any specific way to rank the search results and requires careful thinking of each coefficient which arise from fuzzy comparison. For example, it's needed to set the rules like "which formula is closer to this one – the formula, which contains the query as a subtree or the one which is the same, but with strict comparison signs changed to non-sctrict?" For any imperfect match some penalty may be applied to the final mark of the comparison, but such penalty coefficient need to be fine-tuned in a real-life usage scenarios and after all may depend on personal preference or the domain which formulas used in.

3 Generic problems

The described above methods both have some common problems still to be addressed. First of all it is the contextual analysis, which itself goes far beyond the pure formula search. The context of the formula may contain crucial information like the actual meaning of some predefined constants or functions and its analysis may help in identifying features in the first approach or in constructing of the formula tree for the second enriching the data which could be extracted from the presentational definition of the formula.

Other common problem is the query language. Probably the most accessible way would be the LaTeX math expressions language, but still it remains presentational and in the supplying of the search query the context of the formula is missing, so it may be impossible to determine what exactly from the math semantic point of view does the user mean by saying `J_i(x)`.

4 Clusterization of equations

References

[1] Muhammad Adeel, Hui Siu Cheung, Sikandar Hayat Khiyal, *MATH GO! Prototype of a content based mathematical formula search engine.* Journal of

Theoretical and Applied Information Technology, 2008.

[2] Michael Kohlhase, Ioan Sucan, *A search engine for mathematical formulae.* Springer-Verlag Berlin Heidelberg, 2006.