Scaling Feature Based Mathematical Search Engine for Real-World Document Sets

Jozef Mišutka[1]

Department of Software Engineering, Charles University in Prague
misutka@ksi.mff.cuni.cz

Abstract. There have been several interesting approaches to mathematical searching described in the last few years. We decided to implement another mathematical search engine, building on the work by Ma et al. described in the paper "Feature Extraction and Clustering-based Retrieval for Mathematical Formulas".
We have extended the original algorithms proposed by Ma et al. and implemented them using EgoMath's formula parse trees and applying normalisation and ordering algorithms. The approach has been studied and evaluated on a real-world document set - Wikipedia.org. We introduced several improvements and tested three different versions of the original algorithm with interesting results.

1 Introduction

There have been several interesting approaches to mathematical searching described in the last few years. We decided to implement another mathematical search engine (MSE), building on the work by Ma et al. described in the paper "Feature Extraction and Clustering-based Retrieval for Mathematical Formulas" [Ma+10a]. Ma et al. claim to achieve "promising results" by proposing an "effective feature extraction approach" and supporting their claims with an evaluation done on 884 formulae. We chose the feature based extraction algorithms proposed in the paper because we think that feature based similarity research has potential and it has not been applied to real world document set. We will refer to our implementation of the feature based algorithm search engine as FBA.

We have extended the original algorithms proposed by Ma et al. and implemented them using *EgoMath's formula parse* trees and applying *normalisation* and *ordering algorithms* from the EgoMath mathematical search engine [Mi08b]. The approach has been studied and evaluated on a real-world document set - Wikipedia.org. We introduced several improvements and tested three different versions of the original algorithm with several interesting results.

We will begin with a very brief description of the proposed algorithms by Ma et al. Then, we will describe our contributions to the original algorithms in detail including evaluation and the actual implementation.

2 Original Algorithm

The original proposed solution relies on the Presentation MathML (MML) format which is represented as a tree. There are two types of features extracted from a formula: 1) structural and 2) semantic ones. Semantic features are extracted by traversing the formula tree in preorder. There are operators (operators are represented as <mo> in MML), functions inside identifiers (<mi> MML tag) and nodes (except <mrow> MML tag) that have children containing identifiers (<mi> MML tag) extracted. Structural features are obtained from nodes containing semantic features with depth (in the formula tree) ≥ 1. String obtained by concatenating the names of its parent nodes represents the structural meaning.

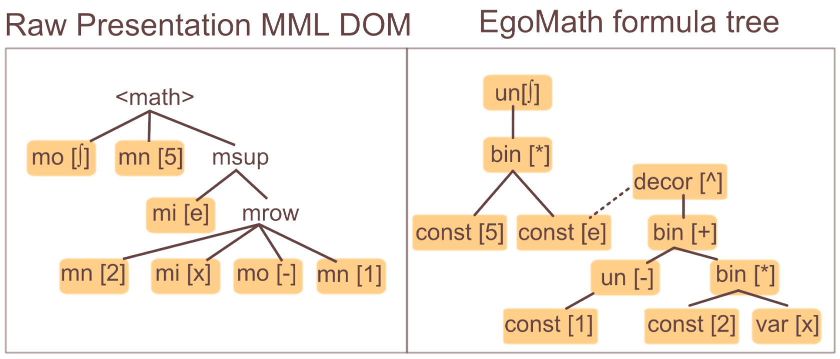

Fig. 1. Different representation of formula trees. The optimistic Ma et al. raw DOM MathML representation is on the left. EgoMath's representation of $\int 5e^{2x-1}$ is on the right (un - unary function, bin - binary function, const - constant, decor - decoration i.e., upper and lower index, var - variable).

Let us consider the formula $\int 5e^{2x-1}$ with MML representation (shown on the left side in Fig. 1) consisting of these tags and their appropriate values mo:$\int, -$, mn:$5, 2, 1$, $mrow$, $msup$, mi:e, x. We obtain these semantic features by using the semantic feature extraction algorithm from the original paper: $\int$ (mo), e (mi + exponential function), $msup$ (identifier in children), $-$ (mo). After extracting the semantic features, we can use the second algorithm to extract the structural features which are ("/" is used as a separator between individual node names): $msup/mrow/-$ ($-$ is a semantic feature and the rest was obtained by concatenating the values by recursively visiting the parents), $msup/e$[1].

The last set of features represents constants and variables. The original paper claims that "Unlike operators or functions, both number constants and variables are not representative, so their semantic features may not be meaningful" and that "It is obvious that exact value representations are not meaningful, e.g.,

[1] This is incorrectly written in the original paper as $e/msup$.

12 or x, for formula search purposes". These are rather strong and subjective claims which we discuss in more detail below. The real values of the numbers and constants are replaced by *var* and *cn* keywords and the same algorithm as for the extraction of the structural features is used. The result features representing constants and variables are $msup/mrow/cn$[2] and $msup/mrow/var$.

Extracted features are transformed into a feature vector using the tf-idf weighting scheme and normalised using cosine normalisation. First, the weights are computed using the term frequency, defined as

$$tf(feature, formula) = \frac{\text{\# of occurences of feature in a formula}}{\max\{\text{\# of occurences of any feature in a formula}\}}$$

and the document frequency defined as

$$idf(feature, formula\_set) = \log \frac{\text{\# of formulae in a formula set}}{\text{\# of formulae which have the particular feature}}.$$

The result vector $\boldsymbol{F}$ is normalised to $\boldsymbol{F}_{norm}$ with

$$\boldsymbol{F}_{norm} = \frac{1}{\sqrt{\sum_{i=0}^{i=n} f_i^2}} \boldsymbol{F}$$

where n is the number of features and f_i is the feature at a particular position. For a small document set, the feature vector can contain all distinct extracted features from all formulae. These normalised feature vectors are used for clustering. The similarity between a query formula and a formula from the document set represented by a normalised feature vector is the well-known *cosine similarity*. The original paper evaluated three different clustering algorithms for retrieval: K-means, self organised maps and agglomerative hierarchical clustering. However, we will not use clustering because it does not directly add to the mathematical awareness. The original evaluation was done on 884 formulae with 20 training and 20 test samples. A more detailed description of the extraction algorithms is available in the original paper [Ma+10a].

3 Modifications

The contribution of the original paper is clear. On the other hand, we must point out several problems we encountered which question the feasibility of the original approach to larger document sets. We try to solve these issues and propose a modified version of the algorithms.

We evaluate the approach on a real-world document set - Wikipedia.org. The Wikipedia.org document set contains formulae encoded in LaTeX. Each formula

[2] Another mistake in the original paper states that $msup/mrow/cn$ was produced by $msup/mrow/5$ but it could only have been produced either by $msup/mrow/2$ or $msup/mrow/1$.

is parsed into the formula tree using the EgoMath library and the *ordering algorithm* is applied. Features are then extracted from the formula tree.

It should also be noted, that the example in the original paper uses an optimistic representation of the original formula $\int 5e^{2x-1}$ in the MML presentation format. Often, automatic converters add more <mrow> elements which make the extracted features more complex and many visually equal formulae have different features. By using normalised EgoMath's formula trees we avoid most of the notation differences which clearly improves the quality in comparison with the original algorithm. The difference between MML representation and the formula tree is illustrated in Fig. 1.

We use Wikipedia.org as the document set for our evaluation and reasoning because it contains common mathematical knowledge and is large enough. Formulae have been extracted using the EgoMath mathematical search engine. The original algorithm produced 271,103 unique features using formula trees over the whole of Wikipedia.org. The precision of this approach on the evaluation queries was very poor (with most of the evaluated queries having near zero P@5[3] precision, see Section 4). The reasons are that variables and constants were considered unimportant. On the other hand, 863,242 features have been extracted when the variables and constants were not replaced by keywords. This was the motivation to introduce limits on the extraction algorithms to lower the number of unique features.

The overall idea is the same as in the original paper. The extraction and concatenation of the names of parent nodes is replaced by a more complex algorithm which: skips binary functions with the same node depth (e.g., $a + b - c$ all leaves have only one parent whose value is extracted), can remove common nodes like $*$, can limit the number of parents traversed and can replace numbers and variables with keywords. 23 out of the 50 most used features contained "*". We added many "*" through ab to $a * b$ normalisation; because of this, and because of the limits set to the feature extraction depth, we added the option to skip "*" nodes when extracting the values from parents.

We define three algorithms with these settings: 1) FBA 1 - skips $*$ from traversing and replaces constants and variables; 2) FBA 2 - skips $*$; 3) FBA 3 - neither replaces any values nor skips $*$.

The final semantic features extracted from $\int 5e^{2x-1}$ by FBA 3 are "$\int$, $*$, e, $\hat{}$ (upper index or msup in MML), $!-$ (unary -), $+$, $*$" (in comparison to "$\int$, e, $\hat{}$, $-$" from the original paper). The final structural features extracted are $int/*$, $int/*/e$, $int/*/e/\hat{}$, $e/\hat{}/+/!-$, $*/e/\hat{}/+$, $e/\hat{}/+/*$, $int/*/5$, $int/*/e$, $\hat{}/+/!-/1$, $e/\hat{}/*/2$, $e/\hat{}/*/x$ (in comparison to $\hat{}/e$, $\hat{}/mrow/-$, $\hat{}/mrow/cn$, $\hat{}/mrow/var$ from the original paper).

The algorithm for extracting numbers and variables visits all nodes and if the node is either a constant, a number or a variable extracts a path similar to the structural extraction. The feature is then appended to the feature vector. In

[3] Precision ($\frac{|\{\text{relevant docs.}\} \cap \{\text{retrieved docs.}\}|}{|\{\text{retrieved docs.}\}|}$) takes all retrieved documents into account, but it can also be evaluated considering only the top n results returned.

contrast to the original algorithm, we also extract features from nodes with a depth equal to 1 because in the formula tree, a root node is part of the formula.

We claim that our changes and usage of EgoMath's formula trees improve the quality of the extraction algorithms in general. We base our claim on several case studies and show several top results of different queries. Firstly, for formulae having only one element, like 5, *sin* would have zero features in the original algorithm but in our approach they have one feature - the node value (or replaced keyword) itself. Mathematically equal (in common mathematical structures) formulae $-2 + a$ and $a - 2$ have different feature sets in the original algorithm but equal feature sets in our algorithm because of the ordering algorithm being applied (the original algorithm would use different MathML representations e.g., additional + sign in the first formula). The same holds for $2 * x$ and $2x$ due to the normalisation algorithms used. Furthermore, 1^x and x^1 have equal feature vectors in the original algorithm but - correctly - different ones in FBA because of the more mathematically precise formula trees. We overcome MML representation which does not respect operator priority, and therefore introduces additional *mrow* elements, by using the formula trees an skipping same priority operators when traversing to the root node.

The main algorithm works with a vector of numbers where each position represents a different feature. Finding the relevant features which should be used to represent every formula in the document set requires either the processing of the whole document set or at least a representative subset. Afterwards, a set of U distinct features is selected and is used to represent each formula.

According to the statistics of the Wikipedia.org document set, 90% (approx. 261,618) of formulae are shorter than 100 characters, including spaces and brackets etc. and 70% are shorter than 50. Despite this fact, the number of unique features using the original feature extraction algorithms over EgoMath's formula trees was in the hundreds of thousands as shown in Fig. 2.

In the first run, we try to reduce the number of unique features by processing only formulae smaller then M characters. Furthermore, the average number of features extracted for the three feature types (structural, semantic and variables and constants) were 8.2, 5.8 and 8.5 respectively which gives the first estimation for the number of the representative feature count. We also limit the maximum number of extracted features for each type by N. Finally, the maximum number of parents visited while concatenating the node names was limited by O. The depth limit set to three means that the formula depth is limited to four nodes in child-parent relations but they must also have different depths (otherwise, they would be skipped). All formulae with a bigger depth have some features stripped. A different depth means that operators on the path to the root node in the formula tree have different priorities.

The number of features extracted with different parameters is depicted in Fig. 2. Finally, we decided to further evaluate $M = 100$, $N = 20$ and $O = 3$ because the number of features considering all FBA algorithms was acceptable compared to the others and was more than double the average number of features. The total number of extracted features using these settings with $\text{FBA}_{100,20,3}$ 3 (means FBA 3 with $M = 100$, $N = 20$ and $O = 3$) was 239,675 but only 28,539 using $\text{FBA}_{100,20,3}$ 1. The $\text{FBA}_{100,20,3}$ 1 number of features (28,539) is significantly lower than the number of extracted features by the original algorithm without bounds (271,103).

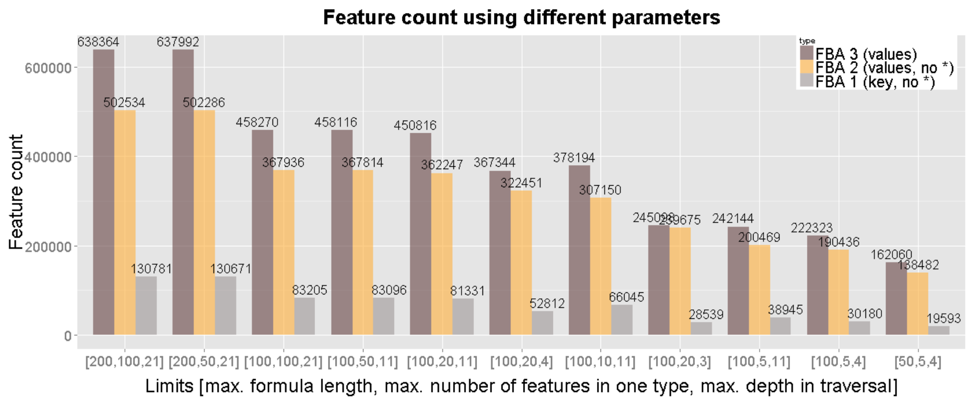

Fig. 2. Number of features using different limits

We take the first 10,000 features sorted by the count of occurrences in the whole document set. It is reasonable to do so because the number of occurrences of the first 30 features dramatically falls and the features become relevant. Starting from the 29^{th} feature (sorted by the number of occurrences), the occurrence count is in the order of thousands. Starting from the 412^{th}, in the order of hundreds and starting from 3346^{th}, in the order of tens. We have experimented with a lower feature count (100, 1000, 5000) but the precision of the result was smaller. Because we focus on applicability we will not consider feature count above 10,000.

4 Evaluation and Results

We will use mathematical documents from Wikipedia.org as extracted using EgoMath mathematical search engine. This document set contains approx. 28,100 mathematical articles with 290,872 unique formulae, out of which, 261,618 are valid for feature extraction having textual length smaller than 100. Out of these, 258,404 have non empty feature sets.

FBA algorithms returned many equal formulae with the same high similarity at the beginning because of the EgoMath normalised parse trees e.g., “\pi = c /d”, “\pi = \frac{c}{d}”. These results are treated as one.

The evaluation was done by performing set of selected queries. Because Wikipedia.org very likely contains many common knowledge subjects, we tried to compile a list of known and famous formulae not restricted by our own research background. We found a web page called “Famous Equations and Inequalities”[4] which we used as the basis. We have evaluated the whole set of queries from the site but selected a smaller set shown below. Top-5 results listed side-by-side are shown below of the original algorithm (but based on EgoMath formula trees), $\text{FBA}_{100,20,3}$ 1 and $\text{FBA}_{100,20,3}$ 3.

Original algorithm	$\text{FBA}_{100,20,3}$ 1	$\text{FBA}_{100,20,3}$ 3
1) $e^{i\pi} = -1$		
$e^{i\pi} = -1$ $e^{\pi} = (e^{i\pi})^{-i} = (-1)^{-i}$ $e^{-t\tau} = e^{-tr^2}$ $\mu = e^{\int p(x)dx} = e^{\int -1dx} =$ e^{-x} $M(x) = e^{\int \frac{-2}{x} dx} =$ $e^{-2\ln x} = e^{\ln x^{-2}} = x^{-2}$	$e^{i\pi} = -1$ $e^{\xi} - 1 = u\xi$ $e^{i\pi} = -1 + 0i$ $\varDelta = e^D - 1$ $r = e^i - 1$	$e^{i\pi} = -1$ $e^{i\pi} = -1 + 0i$ $e^{\xi} - 1 = u\xi$ $e^{i\pi} + 1 = 0$ $DAF = 1 + e^{-c\pi}$
2) $\pi = c/d$		
$w/h = 3.3$ $\frac{a}{a} = 1$ $\frac{Ph}{Laborer} = 1$ $\frac{Ph}{Laborer} = 6$ $div = 0$	$w/h = 3.3$ $\frac{a}{a} = 1$ $dQ/dt{=}0$ $div = 0$ $d/\lambda = 0.6$	$\pi = c/d$ $PI = \frac{mass}{height^3}$ $\pi = \frac{355}{113}$ c/d $\pi = \frac{3927}{1250}$
3) $\frac{d}{dx}e^x$		
$\dfrac{1}{C}e^{\eta_1}$ $e^{x^2}\frac{d^n}{dx^n}(e^{-x^2})$ $\dfrac{1}{C}e^{\eta_k}$ $\dfrac{\frac{n}{m}0}{Span(E)}$	$\dfrac{1}{C}e^{\eta_1}$ $\frac{\lambda^k}{k!} \cdot e^{-\lambda}$ $\frac{e^{-\lambda}\lambda^j}{j!}$ $e^{-\lambda}\frac{\lambda^k}{k!}$ $\frac{e^{-c}c^k}{k!}$	$e^x \sin y e^x \cos y$ $e^x \cos y$ $e^x \sin y$ $S(e^x)$ $E^f \gg E^c$
4) $\sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n$		

[4] Available at `http://www.math.utah.edu/~pa/math/equations/equations.html` (seen March, 2012).

$\sum_{n=0}^{\infty} \frac{m_n t^n}{n!}$ $\sum_{n=0}^{\infty} \frac{f^n(a)}{n!}$ $\sum_j \frac{f_j(x)}{g_j(x)}$ $\sum_R \frac{1}{\dim(R)^s}$ $\sum_{n=1}^{\infty} \frac{1}{a_n x_n}$	$\sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n$ $\sum_{n=0}^{\text{infin}} \frac{f^n(a)}{n!}(x-a)^n$ $\sum_{n=0}^{\infty} \frac{x^n}{n!}$ $\sum_{n=0}^{\infty} \frac{z^n}{n!}$ $\sum_{n=0}^{\infty} \frac{m_n t^n}{n!}$	$\sum_{n=0}^{\text{infin}} \frac{f^n(a)}{n!}(x-a)^n$ $\sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n$ $\sum_{n=0}^{\text{infin}} \frac{f^n(0)}{n!}x^n$ $T(x) =$ $\sum_{n=0}^{\infty} \frac{f^n(x_0)}{n!}(x-x_0)^n$ $f(z) \approx \sum_{k=0}^{\infty} \frac{f^k(c)}{k!}(z-c)^k$
5) $cos^2(x) + sin^2(x)$		
$\cos y + \sin y$ $\cos u + \sin \theta$ $\cos(z) + \sin(z)$ $\cos b + \sin a$ $\sin^2(X) + \cos^2(X) = 1$	$\cos(z) + \sin(z)$ $\sin^2(X) + \cos^2(X) = 1$ $\cos^2 \theta + \sin^2 \theta = 1$ $\sin^2(x) + \cos^2(x) = 1$ $z = \sin^2 x + \cos^2 x$	$\sin^2(x) + \cos^2(x) = 1$ $z = \sin^2 x + \cos^2 x$ $\cos(z) + \sin(z)$ $\cos^2 \gamma + \sin^2 \gamma = 1$ $\cos(t)^2 + \sin(t)^2 = 1$
5) $E = mc^2$		
$\nabla^4 \psi = 0$ $l(x^2) = 2$ $E = \gamma mc^2$ $E = \pm mc^2$ $d\alpha^2 = 1$	$\nabla^4 \psi = 0$ $l(x^2) = 2$ $V = 0.26 \times D^3$ $\nabla^2 p = 0$ $E = \pm mc^2$	$E = mc^2$ $E_0 = mc^2$ $E_0 = m_0 c^2$ $E_{rest} = mc^2$ $E_{\text{rest}} = E_0 = mc^2$
6) $Ax = \lambda x$		
$M = \angle zcy$ $V = V(r)$ $\psi(\alpha) = \delta$ $f(m) = n$ $nZ = (n)$	$M = \angle zcy$ $V = V(r)$ $C = Ab(X)$ $f(m) = n$ $t = O(\epsilon)$	$Ax = \lambda x$ $[A][x] = [x]\lambda$ $y = \lambda x$ $\mathit{\Lambda}(x)$ $Aw = \lambda Bw$

The problematic queries are those which rely on the actual values of elements because there are too many structurally equal ones. This issue is fixed by $\text{FBA}_{100,20,3}$ 2 and $\text{FBA}_{100,20,3}$ 3, and as can be seen, they outperform $\text{FBA}_{100,20,3}$ 1 because they use more relevant features. Only query 3 proved to be problematic because it is too simple and the relevant features were not in the selected 10,000.

The results are surprisingly outstanding for $\text{FBA}_{100,20,3}$ 2 and 3 given the fact that only around 4% of all features were used to characterise each formula. However, we must stress that there can be cases when a formula has important features not present in the first 10,000 features like in query 3 (or even that there is none).

There are several interesting results. Firstly, that variable name are considered unimportant when the structure is similar e.g., $\mathrm{FBA}_{100,20,3}$ 3 in query $\sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n$ with result $T(x) = \sum_{n=0}^{\infty} \frac{f^n(x_0)}{n!}(x-x_0)^n$ or query $cos^2(x)+sin^2(x)$ with results $\cos^2\gamma+\sin^2\gamma = 1$ and $\cos(t)^2+\sin(t)^2 = 1$. The same applies for subscripts e.g., query $E = mc^2$ with results $E_0 = m_0c^2$ and $E_{rest} = mc^2$. It is also interesting that searching for $e^{i\pi} = -1$ returned $e^{i\pi} = -1 + 0i$ in $\mathrm{FBA}_{100,20,3}$ 3.

5 Contribution and Conclusion

We have presented an improved implementation of the feature based search engine originally proposed by Ma et al. which can search for mathematical formulae. We have shown that the original algorithm extracts too many non relevant features and introduced boundaries which limit the number of extracted features. We found out that, for our testing queries, using 4% of all features extracted by $\mathrm{FBA}_{100,20,3}$ 3 was enough for obtaining interesting results on our testing document set. Furthermore, we have shown that relying on EgoMath's formula trees rather than on the Presentation MML makes it possible to find more relevant formulae.

Acknowledegements. This work was supported by the grant SVV-2013-267312.

References

[Ma+10a] Ma K., Hui S. C., Chang K.: Feature extraction and clustering-based retrieval for mathematical formulas. In: Software Engineering and Data Mining (SEDM), pp. 372–377, IEEE, (2010).

[Mi08b] Mišutka J., Galamboš L.: System Description: EgoMath2 As a Tool for Mathematical Searching on Wikipedia.org, LNCS 6824, pp. 307–309, Springer, Berlin Heidelberg (2011).