Combining Effectively Math Expressions and Textual Keywords in Math IR

Giovanni Yoko Kristianto[1], Goran Topić[2], and Akiko Aizawa[1,2]

[1] The University of Tokyo, Bunkyo-ku, Tokyo, Japan
giovanni@nii.ac.jp
[2] National Institute of Informatics, Chiyoda-ku, Tokyo, Japan
goran_topic@nii.ac.jp, aizawa@nii.ac.jp

Abstract

Math IR systems are using mathematical expressions and text in documents to enable full text search. However, effective techniques or weighting schemas to combine math expressions and keywords in search have not yet been fully investigated. This paper examines the effectiveness of several learning-to-rank methods to combine math expression and textual keyword scores. Our experimental results showed that the learning-to-rank methods deliver significantly better performance (up to 25.89% precision improvements) than the standard Lucene tf-idf scoring method.

1 Introduction

In Mathematical Information Retrieval (MIR), users may search for mathematical expressions using queries that contain not only math expressions, but also textual keywords. Majority of the MIR systems have both math and text indexed in their systems in order to handle such queries. However, effective techniques to combine math expressions and textual keywords in math search have not yet been fully investigated. Most MIR systems use either Lucene scoring [8, 9, 12, 16] or linear combination [11, 15] methods. Yet, to use Lucene scoring method as it is may not be optimal, since MIR systems quite often generate more math terms than textual terms from a query, and as a consequence, the score of a matched math expression is mostly composed of scores from math terms. In the systems that use linear combination method [11, 15], each retrieval unit has math- and text-related scores. To obtain a final score for each retrieval unit, these systems linearly combine the math and text scores using predefined weights. In the NTCIR-11 Math-2 Task [1], RIT team [15] reported that weight 0.50 given to the text scores, which means that math and text are equally weighted, achieved the highest ranking performance among other weights (0.00, 0.05, and 0.25). However, due to the limitation of this task, where each participant could submit only up to 4 runs, they had not yet investigated the ranking performance for text weights beyond 0.50. Nevertheless, we suggest that setting the text weight higher than the expression weight (i.e. text weight over 0.50) may deliver better search results than if both text score and expression score have the same weights. This consideration comes from the fact that mathematical expressions, despite their usefulness at expressing concepts, are quite often ambiguous; and such ambiguity can often be resolved by accessing the textual context of each particular expression.

This paper addresses the problem of combining math expressions and text for MIR. We examine the effectiveness of several learning-to-rank (L2R) methods for obtaining the optimal combination. Then, we compare the precision obtained by these methods to the precision acquired by Lucene scoring method, and show that the L2R methods significantly outperform Lucene scoring method. We also investigate the importance of math and text in Math IR.

2 Related Work

In order to make posible searching math using a combination of keywords and math expression queries, MIR research focused on the math expression indexing technique [5, 6, 10, 14] and on how to exploit text. Many previous MIR systems [8, 9, 11, 12, 16], even though they employ different techniques to encode math expressions, use Lucene to index and search both the math and textual keywords. Some other systems [4, 15] employ custom indexing techniques to index the math expressions, but still use Lucene to index the text found in math documents. For a given query containing math and keywords, several systems [8, 9, 12] assign each matched math a score obtained using Lucene scoring method. MIaS [16] uses Lucene scoring function as well, but also adds another parameter to the function, i.e., additional weight for the query term if the term comes from math expressions in the query. Nguyen et. al [13] introduced the Ranking Passive Aggressive algorithm as a method to obtain an optimal weight vector whose each element denotes the weight for an indexed (math or text) term. Another common approach to get a final score of each matched math expression is to linearly combine the score from the math index and the score from the text index [11, 15]. In the RIT system [15], the score for a document is given by linearly combining the maximum score for a math in the document given math query and the score of the document given textual query. They use formula $score(d) = \alpha\ text(d) + (1 - \alpha)\ formula(d)$ to obtain the final score, submitting results for $\alpha = \{0, 0.05, 0.25, 0.50\}$. Their experiment results showed that $\alpha = 0.50$ gave the best precision. Within the works that use Lucene scoring method, the one that adds another weight for each term achieves the highest precision in NTCIR-11 Math-2 Task. Estimating such weights requires either a heuristic approach [16] or a large-scale learning technique [13]. In contrast, the linear combination method with a priori weights can be implemented in a straightforward manner.

To obtain the optimal weights for math and text score in a linear combination, we can apply machine learning method, e.g. multiple linear regression. In fact, learning-to-rank (L2R) refers to machine learning techniques for constructing ranking models. Generally, learning to rank methods can be divided into three categories: pointwise, pairwise, and listwise methods. Pointwise approach, e.g. linear or polynomial regression, transforms the ranking problem into classification, regression, or ordinal classification. The pairwise approach, e.g. RankBoost [3] and LambdaMART [17], transforms ranking into pairwise classification or pairwise regression. The listwise approach, e.g. ListNet [2] and AdaRank [18], takes ranking lists as instances in both learning and prediction. This paper examines the effectiveness of these approaches for combining math and text scores in MIR.

3 Our Approach

Figure 1 shows the general framework of MIR systems. This paper assumes that MIR systems involves two or more indices (or one index, but with two or more fields): at least one for math and one for text. To enable searching using queries that contain math and textual keywords, the MIR systems first perform math expression retrieval and text retrieval using the math and text indices, respectively. This process will generate two or more retrieval lists. The reranking module then combines the scores from these lists to produce a rank list of retrieved math expressions. The implementation of MIR system in this paper used an vertical-path-based technique [7, 8], which encodes the structure and content of math expressions, to store math expressions in a database. There are three fields in our storage schema dedicated to store the encoding results: `opaths` (ordered paths), which stores the vertical path of each node in

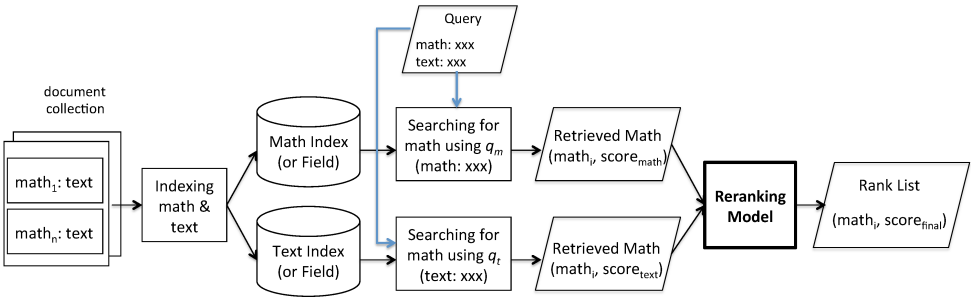

Figure 1: Overview of math search system used in this paper

the math with preserving the ordering information, `upaths` (unordered paths), which stores the same content as `opaths`, but without ordering information, and `sisters`, which stores the sibling nodes in each subtree. We also construct a word vector representation of each math expression's context by associating the math with its surrounding text (10 words before and after the math).

3.1 Combining Expressions and Keywords using Lucene Scoring

We import the encoding result and associated text of each math expression into search platform Apache Solr. In our MIR system, a query can be a combination of math and textual keywords. Query keywords and encoding results of query math are then converted into Solr disjunctive query. We set Solr to utilize default tf-idf similarity, which its practial scoring function is given by (1)[1], to score each math expression f, whose each term is denoted by t, that matches the query.

$$score(q, f) = coord(q, f) \times queryNorm(q) \times \sum_{t \in q} (tf(t, f) \times idf(t)^2 \times norm(t, f)) \qquad (1)$$

The *tf* is the term frequency, *idf* is the inverse document frequency, *norm* encapsulates a few indexing time boost and length factors, *coord* denotes a score factor based on how many of the query terms are found in the specified math, and *queryNorm* is a normalizing factor used to make scores between queries comparable.

3.2 Combining Expressions and Keywords using Learning to Rank

The scores from (1), however, may not be optimal for ranking when the query contains both math and text. For instance, our results for text-only queries are occasionally better than those for queries with both math and text. Our investigation showed that this happens because, in our system, the number of terms generated from math encodings are often much higher than the number of terms from text. This indicates that when we have multiple features for each retrieval units, the default Lucene tf-idf scoring method may not be optimal to combine those features for ranking. Therefore, we need to weight appropriately each feature. Since the proper weights will strongly depend on the implementation of Math IR system, we need a learning method to obtain these weights.

[1] `https://lucene.apache.org/core/5_3_1/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html` (Accessed at: 2015-12-10)

This paper prevents a bias toward math-generated terms by first obtaining two scores, i.e. from math terms and from textual terms, separately. Then, these scores are normalized as $normScore = \frac{rawScore}{1+rawScore}$. The math and text scores are the features for the L2R methods. This paper applies four different L2R methods, which are briefly described as follows. Multiple linear regression estimates the weight of each feature using least square technique. LambdaMART [17] combines the strengths of boosted tree classification and LambdaRank. AdaRank [18] linearly combines weak rankers for making ranking predictions. ListNet [2] uses different probability distributions to define the loss function.

4 Experiments

This paper used the dataset released by NTCIR11 Math-2 Task [1], which consists of 8,301,578 retrieval units (paragraphs) that contain around 60M math expression, 50 topics each of which includes a list of math expressions and a list of keywords, and result of pool assessment (50 retrieval units per topic, manually assessed with a relevancy score 0-4). The relevancy score 0 denotes non-relevant unit, 1-2 partially relevant, and 3-4 highly relevant. In the evaluation, we first generated a ranked list of all retrieval units that match the query. Next, we created a condensed list from the raw ranked list by removing all unjudged retrieval units. We used the condensed list for the evaluation because the number of assessed units per topic is very small compared to the total number of retrieval units in the dataset (incomplete assessment) We evaluated the ranking performance of Lucene's tf-idf and L2R methods for combining math expressions and textual keywords for math searching. For Lucene scoring, the $score(q, p_i)$ of each retrieval unit p_i for a given query q is calculated using step 1.1. in Procedure 1. For L2R methods, since the retrieval unit in our database (math) is different from what the assessment result expects (paragraph), we use Procedure 1 for constructing the training set.

Procedure 1 (Training Set Construction)**.**

1. Get a math expression f_i^* to represent each retrieval unit (paragraph) p_i in the training set.
   1.1. Set $f_i^* = \text{argmax}_{f_{ij} \in p_i} score(q, f_{ij})$ and $score(q, p_i) = score(q, f_i^*)$, where q is a query that contains math and keywords and $score(q = query, f = math)$ is defined by (1).
2. Get features for each retrieval unit p_i in the training set.
   2.1. For each topic, compose math query q_f and textual query q_t.
   2.2. Features: $s_f = score(q_f, f_i^*)$ and $s_t = score(q_t, f_i^*)$
   2.3. Response: binary relevancy score (depends on either high or partial relevancy setting).

For LambdaMART and AdaRank, we set Mean Average Precision (MAP) as the metric to optimize on training data. For testing, the learned models are applied to rank all math found in all retrieval units. We use the step 1.1. in Procedure 1 to obtain $score(q, p_i)$ of each retrieval unit, then rank the retrieval units based on this score. The evaluation metrics are MAP, Precision-at-5, and Precision-at-10. The L2R methods are evaluated using nested cross validation (5-inner-fold for tuning hyperparameters and 10-outer-fold for reporting performance).

Experiment Result

Table 1 shows the ranking performance from L2R, Lucene scoring, and searching using either only math or keywords as queries. First, the L2R methods outperform Lucene scoring

Table 1: Performance from each method.

$^{1-9}$ statistically significant ($p < 0.05$ in ANOVA with post-hoc Tukey HSD) compared to model with the specified ID.

ID	Model	Highly Relevant			Partially Relevant		
		MAP	*P@5*	*P@10*	*MAP*	*P@5*	*P@10*
1.	math ($\alpha = 0.0$)	.4108	.4440	.3820	.5085	.6800	.6220
2.	keywords ($\alpha = 1.0$)	.4953^{1}	.5440^{1}	.5260^{1}	.5876^{1}	.9400^{13}	.9040^{13}
3.	lucene scoring	.5370^{1}	.5760^{1}	.4880^{1}	.6213^{12}	.7840^{1}	.7340^{1}
4.	Mult. linear regr.	**.6259**^{123}	.6520^{12}	**.5660**^{13}	.7471^{123}	.9520^{13}	.9140^{13}
5.	LambdaMART	.6198^{123}	**.6640**^{123}	.5460^{13}	.7478^{123}	.9400^{13}	**.9240**^{13}
6.	AdaRank	.6151^{123}	.6520^{12}	.5500^{13}	.7520^{123}	.9520^{13}	**.9240**^{13}
7.	ListNet	.5913^{12}	.6160^{1}	.5340^{1}	**.7529**^{123}	**.9560**^{13}	**.9240**^{13}
Precision from several text weights α ($score(q, p_i) = \alpha s_t + (1 - \alpha)s_f$)							
$\alpha = 0.50$		.6245	**.6560**	.5660	.7248	.9360	.8720
$\alpha = 0.60$		**.6261**	.6520	**.5720**	.7467	.9520	.9140
$\alpha = 0.80$		.6016	.6320	.5600	.7559	**.9600**	.9240
$\alpha = 0.90$		.5843	.5880	.5520	**.7565**	.9520	**.9280**

Table 2: Weights ($p < 1e - 9$ in two-tailed t-test) for each fold in multiple linear regression.

		Weights for -th fold									
		1	2	3	4	5	6	7	8	9	10
High Relv.	math	.3572	.3564	.3742	.3882	.3909	.3875	.4014	.3891	.3481	.3662
	text	.6428	.6436	.6258	.6118	.6091	.6126	.5986	.6109	.6519	.6338
Part. Relv.	math	.3919	.4005	.3817	.3982	.4010	.4026	.4026	.3993	.3739	.3905
	text	.6081	.5995	.6183	.6018	.5991	.5974	.5974	.6007	.6261	.6095

approach at all metrics. The highest performance obtained by L2R methods improves MAP, P@5, and P@10 by 21.18%, 21.94%, and 25.89%, respectively. Compared to the search that uses only textual keywords as queries, the Lucene scoring method, which combines math and text, surprisingly performs lower, especially at P@10. This happens because, on average, a query generated from each topic contained 94.82 math terms and 3.1 textual terms, and as a consequence, text has a low impact on the score of each retrieval unit. On the other hand, all L2R methods give improvements over text-only search in both relevancy settings. Among the L2R methods, however, there is no statistically significant difference among them. The linear regression method, albeit simple, is quite impressive since it delivers performance close to the highest one. We also examine the importance of text score in comparison to the math score. Table 2 shows the coefficients of math and text scores calculated in linear regression models. The result confirms that (at least in our dataset) text has more impact than math in predicting the relevance of a retrieval unit. The second part of Table 1 reports the highest performance from linearly combining math and text scores using text weight α (from exhaustive weight search). The L2R methods perform comparably to this linear combination with exhaustive weight search. In the linear combination, for high relevancy setting, text is slightly more important than math (the highest precision is at $\alpha = 0.60$). For partial relevancy, $\alpha = 0.90$ gives the highest precision. Such a high α shows that in the dataset, retrieval units that contain dissimilar math expressions, but contain textual terms matching the keywords are likely to be considered as partially relevant. This explains that there is a judgment tendency in partial relevancy cases: text matching is more important than math matching. Yet, there was no explicit statement in the task design of NTCIR-11 Math-2 about this tendency. Such insight should be considered in the task design, so that the participants can tune their systems.

5 Conclusion

We investigate learning-to-rank methods to optimally combine math and text scores for MIR. We show that these methods significantly outperform Lucene tf-idf scoring method. Using MCAT system as backend and NTCIR-11 Math-2 dataset, we find out that text has more impact than math in predicting the relevance of a retrieval unit. For future work, we consider applying several different methods to encode math expressions and extracting more textual information about math expressions, then using L2R to exploit them.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 14J09896, 25245084, and 24300062.

References

[1] A. Aizawa, M. Kohlhase, I. Ounis, and M. Schubotz. NTCIR-11 Math-2 task overview. In *Proc. of the 11th NTCIR Conference*, 2014.

[2] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *Proc. of the 24th International Conference on Machine Learning*, 2007.

[3] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Machine Learning Research*, 4:933–969, 2003.

[4] R. Hambasan, M. Kohlhase, and C. Prodescu. MathWebSearch at NTCIR-11. In *Proc. of the 11th NTCIR Conference*, 2014.

[5] X. Hu, L. Gao, X. Lin, Z. Tang, X. Lin, and J. B. Baker. Wikimirs: A mathematical information retrieval system for wikipedia. In *Proc. of the 13th ACM/IEEE-CS JCDL*, 2013.

[6] S. Kamali and F. W. Tompa. Retrieving documents with mathematical content. In *Proc. of the 36th International ACM SIGIR Conference*, 2013.

[7] G. Y. Kristianto, G. Topić, and A. Aizawa. Exploiting textual descriptions and dependency graph for searching mathematical expressions in scientific papers. In *Proc. of the 9th ICDIM*, 2014.

[8] G. Y. Kristianto, G. Topić, F. Ho, and A. Aizawa. The MCAT math retrieval system for NTCIR-11 Math track. In *Proc. of the 11th NTCIR Conference*, 2014.

[9] P. Libbrecht and E. Melis. Methods to acess and retrieve mathematical content in activemath. In *Proc. of the 2nd ICMS*, 2006.

[10] X. Lin, L. Gao, X. Hu, Z. Tang, Y. Xiao, and X. Liu. A mathematics retrieval system for formulae in layout presentation. In *Proc. of the 37th International ACM SIGIR Conference*, 2014.

[11] A. Lipani, L. Andersson, F. Piroi, M. Lupu, and A. Hanbury. TUW-IMP at the NTCIR-11 Math-2. In *Proc. of the 11th NTCIR Conference*, 2014.

[12] R. Munavalli and R. Miner. Mathfind: A math-aware search engine. In *Proc. of the 29th Annual International ACM SIGIR Conference*, 2006.

[13] T. T. Nguyen, K. Chang, and S. C. Hui. A math-aware search engine for math question answering system. In *Proc. of the 21st ACM CIKM*, 2012.

[14] T. T. Nguyen, S. C. Hui, and K. Chang. A lattice-based approach for mathematical search using formal concept analysis. *Expert Systems with Applications*, 39(5):5820–5828, 2012.

[15] N. Pattaniyil and R. Zanibbi. Combining tf-idf text retrieval with an inverted index over symbol pairs in math expressions: The tanget math search engine at NTCIR 2014. In *Proc. of the 11th NTCIR Conference*, 2014.

[16] M. Růžička, P. Sojka, and M. Líška. Math indexer and searcher under the hood: History and development of a winning strategy. In *Proc. of the 11th NTCIR Conference*, 2014.

[17] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

[18] J. Xu and H. Li. Adarank: A boosting algorithm for information retrieval. In *Proc. of the 30th Annual International ACM SIGIR Conference*, 2007.