# The MCAT Math Retrieval System for NTCIR-10 Math Track

Goran Topić
National Institute of Informatics
Tokyo, Japan
goran_topic@nii.ac.jp

Giovanni Yoko Kristianto
The University of Tokyo
Tokyo, Japan
giovanni@nii.ac.jp

Minh-Quoc Nghiem
The Graduate University for Advanced Studies
Tokyo, Japan
nqminh@nii.ac.jp

Akiko Aizawa
National Institute of Informatics
Tokyo, Japan
aizawa@nii.ac.jp

## ABSTRACT

NTCIR Math Track targets mathematical content access based on both natural language text and mathematical formulae. This paper describes the participation of MCAT group in the NTCIR math retrieval subtask and math understanding subtask. We introduce our mathematical search system that is capable of formula search, and full-text search. We also introduce our mathematical description extraction system which was based on a support vector machine model. Experimental results show that our general-purpose search engine can work reasonably well with math queries.

## Keywords

Mathematical formula search, Description extraction, MathML indexing, Support Vector Machine

## Team Name

MCAT

## Subtasks

Math Retrieval, Math Understanding

## 1. INTRODUCTION

The NTCIR Math Task aims to explore search methods tailored to mathematical content through the design of suitable search tasks and the construction of evaluation datasets. This task consists of three subtask: the math retrieval subtask, the math understanding subtask, and the math free subtask. The math retrieval subtask seeks to retrieve relevant mathematical formulae or documents for a given query and a document collection. The math understanding subtask seeks to extract natural language descriptions of mathematical expressions in a document for their semantic interpretation. The math free subtask seeks contributions on any topics related to the math search and math understanding tasks. In this paper, we describe our participation of MCAT group[1] in the NTCIR Math Task.

The participation of MCAT group focuses on math retrieval and math understanding subtasks of the NTCIR Math Task. For the math retrieval subtask, we built a mathematical search system that is capable of formula search, and full-text search. Search is performed as matching between the factors of the query and the factors in the database, and the results are ranked according to the number of matched factors, modified by Lucene's length normalization and TF/IDF scoring algorithm. For the math understanding subtask, we built a mathematical description extraction system to extract natural language descriptions of mathematical expressions. We train an SVM model using the features extracted from pairs of noun phrases and mathematical expressions in the same sentence. In this paper we describe the submissions for these subtasks and consider how they might be improved.

The remainder of this paper is organized as follows. Sections 2 presents provides a brief overview of the background and related work on math retrieval and math understanding. Section 3 presents our method for the math retrieval subtask. Section 4 describes our method for the math understanding subtask. Section 5 concludes the paper and points to avenues for future work.

## 2. RELATED WORK

### 2.1 Math Search

Current mathematical search systems range from keyword-based, to structure-based and semantic-based systems. Examples includes the Wolfram Functions Site[2], Springer LaTeX Search[3], Uniquation[4], MathWebSearch [1], and MathFind [2]. Yokoi and Aizawa [3] proposed a similarity search method for mathematical expressions that is specifically adapted to the tree structures expressed by MathML. Adeel et al. [4] proposed a content based math search system prototype called the MathGO. These systems offer various math search services using different mathematical markups.

The Digital Library of Mathematical functions project is a mathematical database available on the Web[5]. There are two different approaches for searching for mathematical formulas. In the first approach, it used TexSN textual lan-

[1]http://mathcat.nii.ac.jp/

[2]http://functions.wolfram.com
[3]http://www.latexsearch.com
[4]http://uniquation.com/
[5]http://dlmf.nist.gov

guage to normalize queries and math content to standard forms and allow exact searching. In the second approach, the search system treated mathematical expressions as a document containing a set of mathematical terms. These approaches enabled simultaneous searching for normal text as well as mathematical content.

## 2.2 Description Extraction

There are few studies performed to extract descriptions of mathematical expressions automatically. Regardless of the importance of description extraction of mathematical expressions, previous studies [5] [6] applied relatively naive methods for description extraction. These studies [5] [6] use nouns found in the text around each mathematical formula as descriptions to disambiguate the corresponding mathematical formula. Recent advances in machine learning techniques has brought a significant improvement on information extraction in NLP domain, e.g., in terms of identifying spans in a text given a training data. However, without readily available annotated corpus for training or evaluation, these up-to-date methods were not applicable to this problem. Earlier study [7] applied machine learning method that perform automatic description extraction from Japanese scientific papers, but it considered only the final compound nouns in the preceding noun phrases. Another work [8] introduced annotation design and extraction methods for description. This is an initial work for the improved annotation design provided in the NTCIR Math Understanding subtask [9]. The detail explanation of this annotation scheme is explained by Kristianto et al [10]. This paper contributes to the improvement of previous works by applying better annotation scheme and assuming that descriptions are noun phrases instead of head nouns.

# 3. MATH SEARCH: INDEXING AND SEARCHING

The objective of the math search task was to find relevant mathematical expressions in 100,000 documents regarding three classes of queries: *formula search*, *full-text search* and *open information retrieval* [9].

In formula search, the query is a mathematical expression. Full-text search specifies additional keywords that relate to target expressions. Finally, in open information retrieval, the queries are human-readable questions. In contrast to the first two search types, which were to be fully automated, open information retrieval was to be semi-automated: the results were to be found through a dialogue of the search system and a human operator.

## 3.1 The Problem

MathML defines two different layers: Presentation MathML and Content MathML. The former expresses the layout of symbols used to display a mathematical formula, while the latter encodes its semantics, with no regard to notation. The reason for this split is the fact that mathematical notation is quite inconsistent, and symbol set limited: a notation is commonly reused, and there often exist several different ways of writing down the same core meaning. For example, the derivation of function $y = f(x)$ can be represented as $\frac{d}{dx}f(x)$, $\frac{df(x)}{dx}$, $\frac{d}{dx}y$, $\frac{dy}{dx}$, $f'(x)$, $D_x y$, $\dot{y}$ and more. Conversely, the notation $y(x+1)$ could represent both an application of function $y$, as well as a multiplication of the value $y$ to $x+1$; and the meaning of $i$ in the most common interpretation of $a_i$ is very different from that in $3i+5$. Speaking of the imaginary constant, in fact, there are at least three ways in common use to represent it: some journals and writers use the italic $i$, some the regular i, while Unicode reserves for it the double-struck i at code point 0x214F (`&ImaginaryI;` entity in HTML). Thus, understanding a mathematical equation necessarily involves context: if it is clear from preceding text or formulae that $y$ is a function, it is safe to conclude that $y(x+1)$ refers to application of this function to $x+1$, and not multiplication.

This semantic ambiguity of the Presentation MathML is a large problem for a mathematics search engine. Ideally, one would translate all queries into the content mark-up: we assume the users will care more about the meaning of their query rather than a specific notational form. However, currently there is no method capable of accurately disambiguating the presentation form. While there is ongoing research into this very issue, it is still not at the stage where it could reliably handle any but the simplest mathematical expressions. This ambiguity problem is especially acute when looking at a search query, which will not have the benefit of context.

The inability to reliably extract the meaning of an expression affected our method significantly. The first realization was that we will have to create an index based on the Presentation MathML, which can be obtained, rather than on much more appropriate Content MathML, which can't be. The second realization was that searching for an exact match is unfeasible. If a query looks for $\sum_{i=0}^{n} a_i$, exact matching will not find $\sum_{i=0}^{n} a_i$ — even though both are readily understood to be the same thing by a human reader. Similarly, a query for the relationship between velocity and acceleration $v = at$ would not find $v = gt$, a special case where acceleration of gravity is written as $g$ instead of the usual $a$. Ideally, $v = gt$ would still be found, but would be ranked lower than the perfect match, $v = at$.

Therefore, the method needed to be flexible in order to account for the notational differences, in some ways similar to full-text search, yet still encode the structural information necessary to distinguish $x\frac{y}{z}$ from $\frac{x}{y}z$. We also wanted to be able to search by context, a very important requirement in the current task, which would require a full-text search engine. The challenge then is how to incorporate mathematical search with full-text search.

## 3.2 Our Method

The full-text search requirement was solved easily, by adopting Apache Solr [11], the most popular open-source full-text search engine. Starting from this choice, we used the method described below to index and search the mathematical formulae, and thus provide a flexible, soft-matching, unified interface for different query types (both formula search and full-text search).

In our Solr schema, the multi-valued `description_XX` fields hold any natural language text associated with a given expression, where `XX` is a language identifier. There are three fields dedicated to encoding the structure and content of a mathematical expression — `opaths`, `upaths` and `sisters` — which are described below in detail. Finally, there are also various other fields serving as keys: the primary key field `gmid`, the foreign key `gpid` identifying the document where

the expression is found, and the key `gumid` which identifies different appearances of the same expression within a single paper as a single class.

Each expression, both at index time and at query time, is transformed into a sequence of keywords across several fields. First, the XML expression tree is cleaned up a little, in an attempt at basic normalization: it is run through SnuggleTeX [12] up-conversion semantic enrichment module, and then unnecessary `mrow` and `mfenced` nodes are removed (since they provide no semantic information other than hierarchy). Next, vertical paths are gathered into the `opaths` (ordered paths) field in such a way that ordering is preserved. In some cases (such as looking for $b+c$ and trying to match $a+b+c$), ordered paths will not be effective, so we introduce the `upaths` (unordered paths) field, with exactly the same information as in `opaths` but with ordering information removed. This vertical path encoding is performed not only for the expression's tree, but for each of its subtrees as well, to achieve a hit on $a(b+c)$ for the query $b+c$. The third and final field carrying the expression structure, `sisters`, lists the sister nodes in each subtree.

```
<math>
    <mrow>
        <msubsup>
            <mo>&Sigma;</mo>
            <mrow>
                <mi>i</mi>
                <mo>=</mo>
                <mn>0</mn>
            </mrow>
            <mi>n</mi>
        </msubsup>
    </mrow>
    <msub>
        <mi>a</mi>
        <mi>i</mi>
    </msub>
</math>
```

**Figure 1:** MathML example: $\sum_{i=0}^{n} a_i$

Figure 1 presents a simple mathematical formula, $\sum_{i=0}^{n} a_i$, written in MathML. Figure 2 shows an example of its representation our index. The first `opaths` encodes the whole MathML tree. The root of the whole tree, an `<mrow>`, is not shown — as explained above, `mrow` has no semantic value, and so do not explicitly list it in the index. The root has two children: `<msubsup>` and `<msub>` elements, which is reflected in `opaths:1#msubsup 2#msub`: the first top-level child being `<msubsup>`, and the second `<msub>`. Grandchildren are given similarly, their position at each level being specified by consecutive numbers, divided by separators. If the element is one of the leaf elements (`<mo>`, `<mn>` or `<mi>`), it also has an additional separator and the representation of its value. Thus, `opaths:1#2#3#mn#1` says that the third child of the second child of the first top-level element is a number with a value of 1 (i.e. `<mn>1</mn>`).

The second row does the same for the sub-expression $\sum_{i=0}^{n}$; the third for the subscript $i = 0$; and the fourth for $a_i$. The sub-expressions $n$, $i$, 0 and $a$ have no structure to encode, so they do not have their own `opaths`.

**opaths:**
`1#msubsup 1#1#mo#Σ 1#2#1#mi#i 1#2#2#mo#= 1#2#3#mn#1 1#3#mi#n 2#msub 2#1#mi#a 2#2#mi#i`

**opaths:**
`msubsup 1#mo#Σ 2#1#mi#i 2#2#mo#= 2#3#mn#1 3#mi#n`

**opaths:** `1#mi#i 2#mo#= 3#mn#1`

**opaths:** `msub 1#mi#a 2#mi#i`

**upaths:**
`#msubsup ##mo#Σ ###mi#i ###mo#= ###mn#1 ##mi#n #msub ##mi#a ##mi#i`

**upaths:** `msubsup #mo#Σ ##mi#i ##mo#= ##mn#1 #mi#n`

**upaths:** `#mi#i #mo#= #mn#1`

**upaths:** `msub #mi#a #mi#i`

**sisters:** `mi#i mo#= mn#1`

**sisters:** `mo#Σ mi#n`

**sisters:** `mi#a mi#i`

**sisters:** `msubsup msub`

**Figure 2:** Encoding example: $\sum_{i=0}^{n} a_i$

As said above, `upaths` is almost the same, but with ordering information removed. Accordingly, `upaths:###mn#1` just says that a child of a child of a top-level element is the number 1.

Because the ordering information is removed for all levels, not just the lowermost one, in a contrived example of $\frac{a+b}{c+d+e}$ and $\frac{c+a}{b+e+d}$, they will both match equally for the query $\frac{x}{e+c+y}$ (where the latter is a plausible match, with substitutions $x = a + b$, and $y = d$). This situation comes about because `opaths` are not applicable to any element ($e$ is never first in its subtree, $c$ is never second, and $x$ and $y$ do not match at all); while `upaths` won't differentiate between sub-elements of the numerator and the denumerator, and thus equally find both $e$ and $c$. To solve this, we introduce the field `sisters`, whose role is to group sister elements together, and in this case boost the score of the first expression where both $e$ and $c$ are found in the same subtree, over the second expression where they aren't.

Matching is then performed by a normal Solr disjunctive query (using default query parser). It considers the factors in the query as independent, then modifies the score using TF/IDF and length normalization.

The result is a very flexible, heuristic system that ranks the search results by similarity to the query expression, and to a degree avoids the pitfalls of inconsistent representation. The downside is that it is *too* flexible: it is difficult to say where the relevant results stop and random matches begin; thus we predict higher recall, but lower precision rates than exact match systems.

## 3.3 Results

We submitted results for all topics in the two categories we participated in: formula search and full-text search. We did so for two runs: one using the original topics as specified by the task, unchanged (MCAT.org); the other with several minor edits to topics to fit the syntax and operators of Lucene full-text search, with no change to content (MCAT.mod).

Table 1 shows the summary of our submission results for the second run. By the metric prescribed by the task, where any topic without a submitted answer scores 0, when compared with other participants' scores as listed in [9], our algorithm came out on top in both categories. Looking at MAP average, when only accepting fully relevant results, our score was 0.162 (compared to 0.127 of the second-placed result) in formula search. In full-text search, we scored 0.297 (with the other participant scoring 0.020). Our scores were even better when partially relevant results were accepted: our score for formula search was 0.379 (next best score being 0.144), and for full-text search we had 0.534 (compared to 0.042 scored by the other participant). We did not submit any results for the open information retrieval category.

**Table 1: Summary results of Math Search subtask**

| | P-10 avg | P-5 avg | MAP avg | Precision |
|---|---|---|---|---|
| Formula Search | | | | |
| Relevant | 0.229 | 0.219 | 0.162 | 0.065 |
| Partially Relevant | 0.500 | 0.476 | 0.379 | 0.220 |
| Fulltext Search | | | | |
| Relevant | 0.293 | 0.320 | 0.297 | 0.103 |
| Partially Relevant | 0.660 | 0.680 | 0.534 | 0.309 |

Since we submitted the results for all topics, regardless of how bad they were, ignoring the unanswered topics brings the scores of several other participants in the formula search subtask above ours.

# 4. DESCRIPTION EXTRACTION

The goal of Math Understanding Subtask is extracting natural language descriptions of mathematical formulae in a document [9]. This section explains method applied to tackle this task.

## 4.1 Our Method

A baseline method is applied in this subtask by naively assuming that nouns found in the apposition of mathematical expressions are descriptions. A descriptions extracted by this method might be a combination of a determiner, an adjective, and nouns. For example, let consider a sentence of "...there exists a point $y \in F^n$ such that ..." from [13]. The baseline method determines text of "a point" as the description of "$y \in F^n$" because it has POS tags of "DT NN" and it appears as an apposition of the expression.

This paper proposes a machine learning model to extract descriptions automatically. The general steps conducted in this method depicted in Figure 3. There are several preprocessing steps in description extraction process. Sentence-splitting step is first applied to each scientific paper using sentence splitter implemented by brat annotation tool [14], then Stanford sentence parser [15] is used to generate a parse tree for each sentence, and finally a set of noun phrases is obtained by analyzing the parse trees. There is an assumption during the extraction process, that descriptions are noun phrases. Therefore, the noun phrases obtained from the parse tree are candidates for being descriptions of the mathematical expressions that appear in the same sentence. A support vector machine (SVM) model trained using linear kernel is used to predict which noun phrases are the descriptions of each particular mathematical expression. The training and predicting processes are conducted using libSVM [16] and Weka [17].

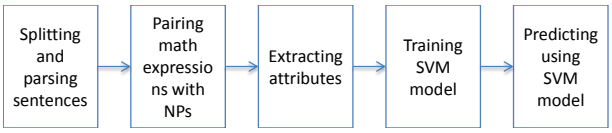


Figure 3: Steps of experiment

The set of features shown in Table 2 is used by the SVM model during training and predicting process. For instance, let consider the sentence of "Let $F$ denote a finite field of $q$ elements" from [13]. One of the noun phrases found in the sentence is "a finite field of $q$ elements". Considering this phrase to be a description of $F$, the extracted attributes are as follows: the noun phrase does not appear as apposition and there are no colon, comma, and other mathematical expressions between the noun phrase and target mathematical expression $F$, therefore the values of attributes 1-4 are all false. Moreover, the noun phrase is not enclosed by parentheses, so the attribute 5 is also false. Furthermore, the noun phrase appears after $F$ and there is one word between them, thus the values of attributes 6 and 7 are 1 and true (means description candidate appearing after target mathematical expressions), respectively. In addition, two tokens appearing before the noun phrase are "MATH" (MATH) and "denote" (VBP), and one token appears after it: "." (.). This attribute is combined with information about the first and last word of the noun phrase: "a" (DT) and "elements" (NNS) Moreover, there is a token appearing before $F$: "let", and three tokens appear after it: "denote", "a", and "finite". Finally, unigram, bigram, and trigram are applied to these tokens surrounding the noun phrases and surrounding the $F$, and produce features: "denote|a", "VBP|DT", "a|finite", "DT||JJ", "denote|a|finite", "VBP|DT|JJ", "MATH|denote", "MATH|VBP", "MATH|denote|a", "MATH|VBP|DT", "elements|.", and "NNS|.". The last extracted attribute is the first verb appearing between $F$ and the noun phrase: "denote".

## 4.2 Experiment

There are 35 papers in the annotated dataset and 10 papers in the evaluation dataset. There are four SVM models built for the experiment: two models predicting full descriptions and two models predicting short descriptions. The result of each model is submitted separately as a different run. The evaluation of the results is depicted in Table 3. The MCAT_full_1 and MCAT_short_1 runs implement all attributes in Table 2 for predicting full descriptions and short descriptions, respectively. On the other hand, MCAT_full_2 and MCAT_short_2 implement only attributes 2-12.

Evaluation result depicted by Table 3 shows that SVM models outperform the baseline method in all evaluation scenarios. The precision of machine learning method is obviously better than baseline since it examines attributes of each noun phrase to decide if it is a description, while the baseline naively assumes all appositive nouns of expressions to be descriptions. Furthermore, machine learning method considers the possibility of all noun phrases in sentences to be descriptions, while the approach taken by baseline

**Table 2: List of Machine Learning Attributes**

| No. | Feature |
|---|---|
| 1 | Test if description candidate appears as apposition of the target mathematical expression |
| 2 | Test if there is a colon between description candidate and the target mathematical expression |
| 3 | Test if there is a comma between description candidate and the target mathematical expression |
| 4 | Test if there is other mathematical expressions between description candidate and the target mathematical expression |
| 5 | Test if description candidate is inside parentheses and target mathematical expression is outside parentheses |
| 6 | Word-distance of description candidate from target mathematical expression |
| 7 | Position of description candidate relative from target mathematical expression (after or before) |
| 8 | Surface text and POS tag of two previous and next tokens around description candidate |
| 9 | Surface text and POS tag of the first and last tokens of description candidate |
| 10 | Surface text and POS tag of three previous and next tokens around target mathematical expression |
| 11 | Unigram, bigram, and trigram of attributes 10 and 8 that is combined with 9 |
| 12 | Surface text of the first verb that appears between description candidate and target mathematical expression |

method overlook this possibility. Therefore, the recall performance of machine learning method is better than the baseline.

Performance comparison of the first SVM model (MCAT_full_1 and MCAT_short_1) with the second one (MCAT_full_2 and MCAT_short_2) indicates that the feature of apposition slightly improve the performance of the extraction of short descriptions. On the other hand, this feature slightly decrease the performance of full descriptions extraction. These two findings demonstrate that apposition feature is good for predicting short descriptions, but not for full descriptions. From analysis result, it is found that textual information appearing as apposition in most cases can be considered as both a complete short description and a part of full description. For instance, let consider text of "...there exists a homogeneous degree $d$ polynomial $g \in F[x_1, \ldots, x_n]$ such that $g$ is not the zero polynomial and $\forall x \in K, g(x) = 0$" from [13]. The apposition information of mathematical expression $g \in F[x_1, \ldots, x_n]$ is "a homogeneous degree $d$ polynomial". This apposition is annotated as a short description and as a part of full description "a homogeneous degree $d$ polynomial such that $g$ is not the zero polynomial and $\forall x \in K, g(x) = 0$". Therefore, this apposition information is considered as correct short description, but incorrect full description during training process. Hence, the apposition information tends to be rejected during full description prediction, but be accepted during short description prediction. This explains why the existence of apposition feature increasing the prediction performance of short descriptions, but not in the case of full descriptions.

**Table 3: Evaluation Result of Baseline and Machine Learning Methods**

| Run ID | Precision | Recall | F-1 |
|---|---|---|---|
| Strict Matching Evaluation | | | |
| baseline_full | 44.10 | 23.85 | 30.96 |
| full_(all noun phrases) | 2.70 | 64.17 | 5.18 |
| MCAT_full_1 | 61.94 | 37.03 | 46.35 |
| MCAT_full_2 | 61.92 | 37.33 | 46.58 |
| baseline_short | 55.35 | 29.94 | 38.86 |
| short_(all noun phrases) | 3.18 | 75.35 | 6.10 |
| MCAT_short_1 | 68.24 | 40.42 | 50.77 |
| MCAT_short_2 | 67.67 | 40.22 | 50.45 |
| Soft Matching Evaluation | | | |
| baseline_full | 64.21 | 34.73 | 45.08 |
| full_(all noun phrases) | 7.30 | 89.12 | 13.49 |
| MCAT_full_1 | 86.48 | 47.41 | 61.24 |
| MCAT_full_2 | 87.25 | 48.30 | 62.18 |
| baseline_short | 64.21 | 34.73 | 45.08 |
| short_(all noun phrases) | 5.82 | 87.33 | 10.91 |
| MCAT_short_1 | 81.68 | 42.81 | 56.18 |
| MCAT_short_2 | 81.24 | 42.61 | 55.90 |

In addition, further analysis shows that by considering all noun phrases as descriptions, there are 64-75% and 87-89% of total descriptions can be obtained by strict matching and soft matching scenarios, respectively. These values indicates the possible highest recall performance for the experiments. Therefore, the recall performances from four runs are in the range of 48.79-57.71%. The low recall performance indicates that current SVM model is not accurate enough to do the prediction. Extraction of more advanced features is required to reduce the bias of the current model. Information from dependency trees that is generated during sentence parsing process can be considered as possible features.

## 5. CONCLUSION

In this paper, we have presented MCAT's submissions to the NTCIR Math Task. For the math retrieval subtask, we have introduced `opaths`, `upaths` for indexing and a modified TF/IDF score for ranking. For the math understanding subtask, we have proposed an SVM classification to detect descriptions of mathematical expressions. Although our work is still at a preliminary stage, the results showed that a general-purpose search engine can work reasonably well with math queries. Meanwhile, our work in the math understanding subtask demonstrates that the SVM classification models outperform baseline method.

This research has raised many questions in need of further investigation. We plan to extend our method using the current system as a baseline. Potential improvements include the following: (1) Normalization of commonly interchangeable MathML elements. (2) Implementation of common subexpression unification rules, which would additionally penalize the results where the instances of the same subexpression are replaced by different subexpressions. (3) Restriction of the number of disjunct factors, since their number adversely impacts search times. (4) Extraction of more advanced features for the math understanding subtask, such as information from dependency trees.

## Acknowledgments

Work partially supported by Kakenhi, MEXT Japan [24300062].

## 6. REFERENCES

[1] M. Kohlhase and I. Sucan. A search engine for mathematical formulae. *Artificial Intelligence and Symbolic Computation Lecture Notes in Computer Science Vol. 4120*, pages 241–253, 2006.

[2] R. Munavalli and R. Miner. MathFind: a math-aware search engine. In *ACM SIGIR conference on Research and development in IR*, pages 735–735, 2006.

[3] K. Yokoi and A. Aizawa. An approach to similarity search for mathematical expressions using MathML. In *2nd Workshop Towards a Digital Mathematics Library*, pages 27–35. DML 2009, 2009.

[4] M. Adeel, H.S. Cheung, and S.H. Khiyal. Math GO! Prototype of a content based mathematical formula search engine. *Journal of Theoretical and Applied Information Technology Vol. 4, No. 10*, pages 1002–1012, 2008.

[5] M. Grigore, M. Wolskam, and M. Kohlhase. Towards context-based disambiguation of mathematical expressions. 2009.

[6] M. Wolska, M. Grigore, and M. Kohlhase. Using discourse context to interpret object-denoting mathematical expressions. pages 85–101, 2011.

[7] K. Yokoi, M-Q. Nghiem, Y. Matsubayashi, and A. Aizawa. Contextual analysis of mathematical expressions for advanced mathematical search. In *CICLing*, 2011.

[8] G.Y. Kristianto, M-Q. Nghiem, Y. Matsubayashi, and A. Aizawa. Extracting definitions of mathematical expressions in scientific papers. In *JSAI*, 2012.

[9] A. Aizawa, M. Kohlhase, and I. Ounis. Overview of The NTCIR 10 MATH pilot task. In *The NTCIR 10*, 2013.

[10] G.Y. Kristianto, M-Q. Nghiem, N. Inui, G. Topić, and A. Aizawa. Annotating mathematical expression definitions for automatic detection. In *Mathematics Information Retrieval 2012 Workshop*, 2012.

[11] Apache Software Foundation. Apache Solr v4.1, 2013. http://lucene.apache.org/solr/.

[12] D. McKain. SnuggleTeX, 2011. http://www2.ph.ed.ac.uk/snuggletex/.

[13] Z. Dvir. On the size of Kakeya sets in finite fields. 2008.

[14] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, 2012.

[15] D. Klein and C.D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the ACL*, pages 423–430, 2003.

[16] C-C. Chang and C-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–27, 2011.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software: An update. In *SIGKDD Explorations*, volume 11, 2009.