博士論文（要約）

Retrieval and Disambiguation of Mathematical Expressions for Mathematical Information Access

（数学情報アクセスのための数式表現の検索と曖昧性解消）

Giovanni Yoko Kristianto

クリスティアント　ギオヴァニ　ヨコ

Retrieval and Disambiguation of Mathematical Expressions for Mathematical Information Access

数学情報アクセスのための数式表現の検索と曖昧性解消

by

Giovanni Yoko Kristianto

クリスティアント　ギオヴァニ　ヨコ

A Doctor Thesis

博士論文

Submitted to

the Graduate School of the University of Tokyo

on December 9, 2016

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Information Science and Technology

in Computer Science

Thesis Supervisor: Akiko Aizawa　相澤彰子

Professor of Computer Science

ABSTRACT

Mathematical expressions are important for communication of scientific information, for instance, to give formal definitions of concepts written in natural language. In this dissertation, we propose a mathematical information access (MIA) system which can help people access and understand math expressions in scientific documents. MIA has been studied in the digital mathematics library and information retrieval communities for searching for math expressions based on their token elements (e.g. identifiers, numbers, and operators) and structures (e.g. fractions, scripting, and matrices). The major focus of the existing work on MIA has been the development of algorithms to store the tree representation of math expressions into a database. On the other hand, the descriptive text of the math expression (hereinafter, we call it textual information) has not yet fully exploited for MIA. The use of textual information has potential to improve the retrieval performance of an MIA system and helps the MIA user understand the definitions of math expressions.

This dissertation presents a framework of MIA that supports math search and math understanding of the users by utilizing math structure and text similarities. We introduce three core ideas which are essential for realizing MIA.

The first core idea in this dissertation is the development of a math information retrieval (MIR) system that exploits the structures and the textual information of math expressions to allow effective search for mathematical knowledge. Following the convention of the current digital math library research community, we assume that a query is given as the combination of math expressions and textual keywords. Our proposed math search system takes advantage of multiple types of textual information to enable high-recall and high-precision retrieval. An evaluation in NTICR-12 MathIR, a mathematics information retrieval shared task, shows that our search system achieved the best performance over other existing math search systems in retrieving highly relevant paragraphs containing the users' requested math expressions.

The second core idea in this dissertation is the enrichment of the textual information of a math expression considering the relationships with other math expressions within the same document. The motivation behind is that textual descriptions of the component subexpressions or identifiers are useful to explain a complex math expression, yet these descriptions may not be captured within the context of the target expression. Therefore, to enrich the textual information of each math expression, while keeping its capability to enable high precision search, we utilize the dependency relationships (e.g. formulae-variables relationships) between math expressions. An evaluation shows that this approach has a significant impact in improving search precision.

In addition to the development of math search system, this dissertation formulates a task of determining the identity of math expressions in documents by linking these expressions to their corresponding entities in knowledge base, such as Wikipedia. This task is denoted as math entity linking (MEL). We propose a supervised learning based approach using math related features, such as math and text similarities, as well as the

location and importance of the math expression within the document. Our evaluation shows that the proposed approach can determine correct links for math expressions in a higher precision than a straightforward application of an MIR system.

To conclude, this dissertation proposes the use of math structures and multiple types of textual information incorporated with the dependency relationships between math expressions to capture the semantics inherent in math expressions. The proposed math search system shows the highest search performance among other four state-of-the-art search systems. In addition, we propose a MEL module reliable enough to link math expressions to their best non-null corresponding entities in knowledge base. Since math expressions are essential part of scientific information, our proposed approach has an important implication for the applications of information access in a wide range of scientific fields. Finally, this dissertation is a step towards enabling effective formula search and formula browsing in digital library practices.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Mathematics plays a fundamental role in science, technology, and engineering. As a consequence, scientific documents or publications frequently contain mathematical knowledge, which is presented in the form of mathematical expressions or formulae. These math expressions are essential for communicating information in the scientific documents — for instance, to explain or define concepts written in natural language. Regarding the number of documents containing math expressions, there are 120,000 journal articles per year in pure/applied mathematics In addition, there are 1,208,910 e-prints stored in arXiv[1] up to November 2016. These include papers in the fields of Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance, and Statistics. Furthermore, it was estimated that the cumulative total of scholarly research articles in existence passed 50 million in 2009 (Jinha, 2010) with a doubling time of 8-15 years (Larsen and von Ins, 2010). In spite of the importance of math expressions and the large number of documents containing them, current general-purpose search engines such as Google cannot effectively locate math expressions contained in a scientific document (Zanibbi and Blostein, 2012).

A mathematical information access (MIA) system is required to help people effectively locate and understand math expressions contained in documents. In this dissertation, we propose an MIA system that consists of two modules, namely

- **Math Search**, which allows people to search for math expressions in a document collection.
- **Math Entity Linking (MEL)** for document browsing. A document browser shows the user the document containing the requested math expressions. This browser is expected to be a reading assistant for the user

[1] `https://arxiv.org/`

by providing information about math expressions contained in each document. For this purpose, the MEL module attempts to link/ground math expressions in the documents to their corresponding entities in a knowledge base (which is Wikipedia in this dissertation).

Figure 1.1 illustrates the process of these two modules. For instance, a person wants to know the applications of the concept of "information theoretic entropy," which is usually denoted by $H(X)$. To do so, they specify the math expression $H(X)$ and "entropy" as the query. In response, the math search module of the MIA system retrieves a ranked list of math expressions and their textual meanings. These math expressions exist in the document collection that is indexed by the MIA system prior to any user query. Once the person selects one of the returned math expressions, the MIA system connects them with the document containing the selected expression. Then, the MEL module of the MIA system identifies the math expressions in the document, i.e., $D(P||Q) = \sum_{x \in S} P(X) log \frac{P(x)}{Q(x)}$ and $H(X) = -\sum_{x \in S} P(X) log P(X)$. Subsequently, this module provides information about these expressions by linking each of them to the corresponding entity in Wikipedia — for instance, the former expression to the Wikipedia article of "Kullback–Leibler divergence" and the latter one to the "Entropy (information theory)."

1.2 Objective and Challenge

This dissertation is dedicated to the development of an MIA system that includes math search and MEL modules. This system is expected to enrich current digital library practices by enabling mathematics-aware information retrieval (MIR) and the grounding of mathematical knowledge in documents.

The main challenge when developing accurate math search and MEL modules is to address the ambiguity inherent in math expressions. Most of the previous work on math search systems attempted to capture the semantics of the math expressions from their structures. They focused on establishing techniques to index the structures of math expressions. However, we suggest that the ambiguity of math expressions cannot be solved by only capturing the structures of the expressions. There are three characteristics of math expressions (Kohlhase and Sucan, 2006) that we consider to be the cause of this ambiguity: (1) mathematical notation is context-dependent (i.e., identical mathematical presentation can represent multiple distinct mathematical objects), (2) different mathematical notations may actually mean the same, and (3) certain variations in notation are

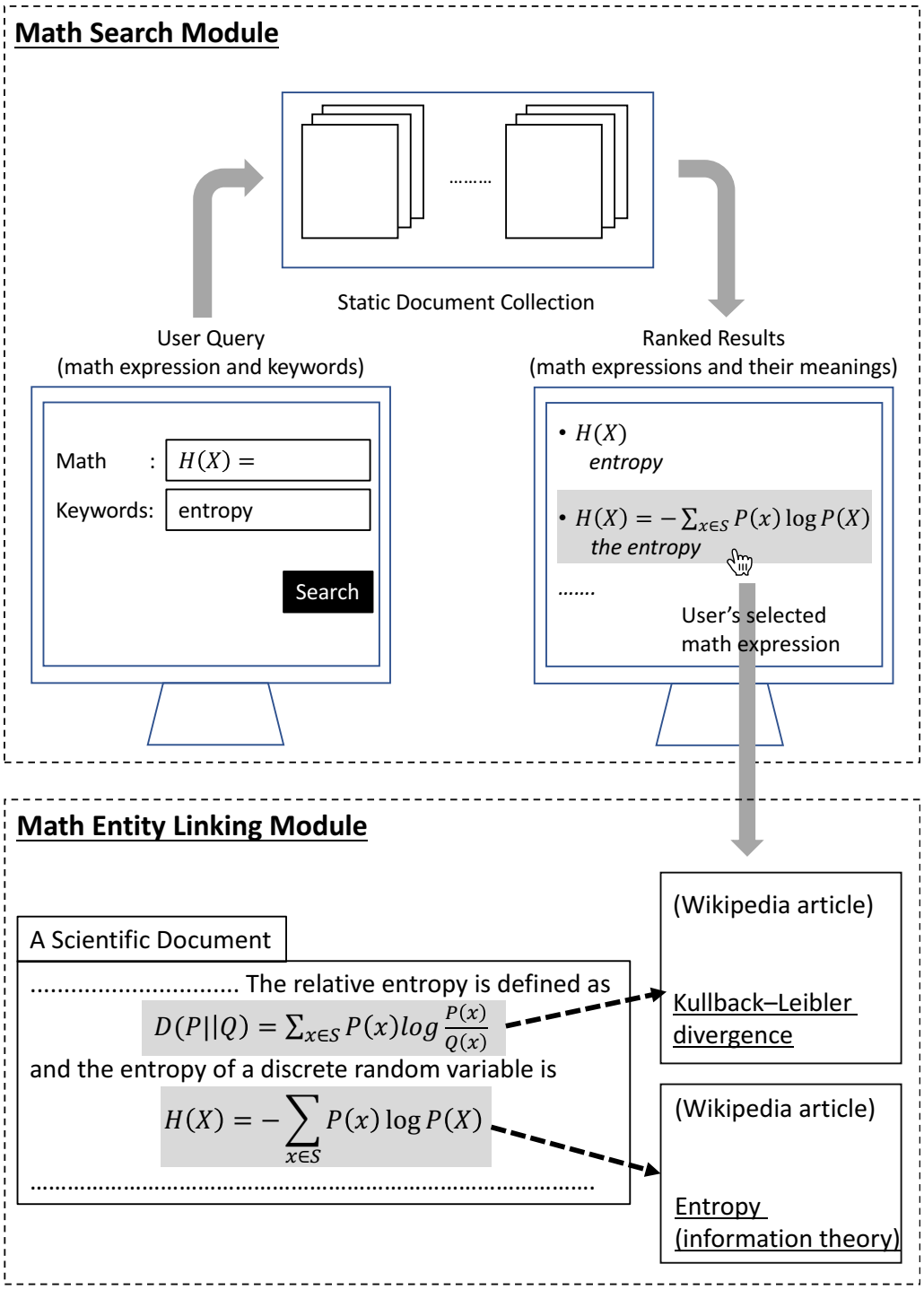


Figure 1.1: Mathematical Information Access system

widely considered to be irrelevant.

We attempt to overcome these problems by capturing the meanings of math expressions through their textual information. In this dissertation, we extract two categories of textual information for each math expression in the document collection, namely

- Textual information explaining the target math expression.

For instance, the mathematical expressions ${}_nP_k$, ${}^nP_k$, $P_{n,k}$, and $P(n,k)$ represent the same mathematical concept, namely k-permutations of n, even though they have different representations. The availability of textual information for each of these expressions is intended to assist the MIA system to determine that these expressions refer to the same concept. In this case, the textual information is expected to contain at least the key term "permutation."

- Textual information that explains the meaning of each of the symbols contained in the target math expression.
  For instance, the left-hand part of Fig. 1.2 indicates that the math expression $\frac{Q}{S}$ may express the same quantity as $\frac{Q}{A}$, given that, in the examined documents, Q has the same textual information (i.e., "total electric charge") in both expressions, and S and A can both be used to express "surface area". Thus, the textual information of S and A should also help disambiguate $\frac{Q}{S}$ and $\frac{Q}{A}$. Conversely, the right-hand part of Fig. 1.2 shows that the same expression $\frac{Q}{S}$ is used both for "surface charge density" and "superficial velocity," depending on whether Q represents the "total electric charge" or "volume flow rate." Again, the textual information of the sub-expression Q helps to determine the meaning of the larger expression.

By doing so, we expect our search module to identify whether two math expressions have the same meaning, and our MEL module can link each math expression to the correct Wikipedia article.

1.3 Problem Statement

Even though we have already identified that the textual information of math expressions and their constituent symbols will help us address the main challenge in developing an MIA system (i.e., ambiguity of math expressions), we still need to determine how to extract this textual information and how to capture the dependency relationships, e.g., formulae-symbols relationships, between math expressions. Moreover, for the math search module, we need to design an effective technique to combine both the textual information and the information regarding the structures of math expressions. Furthermore, for the MEL module, we need to devise a strategy to precisely link math expressions to their corresponding Wikipedia articles. As we will see, the information we use in the math search module is not sufficient for constructing a reliable MEL module. Here, we concisely describe the development process of our MIA system: (1) framework

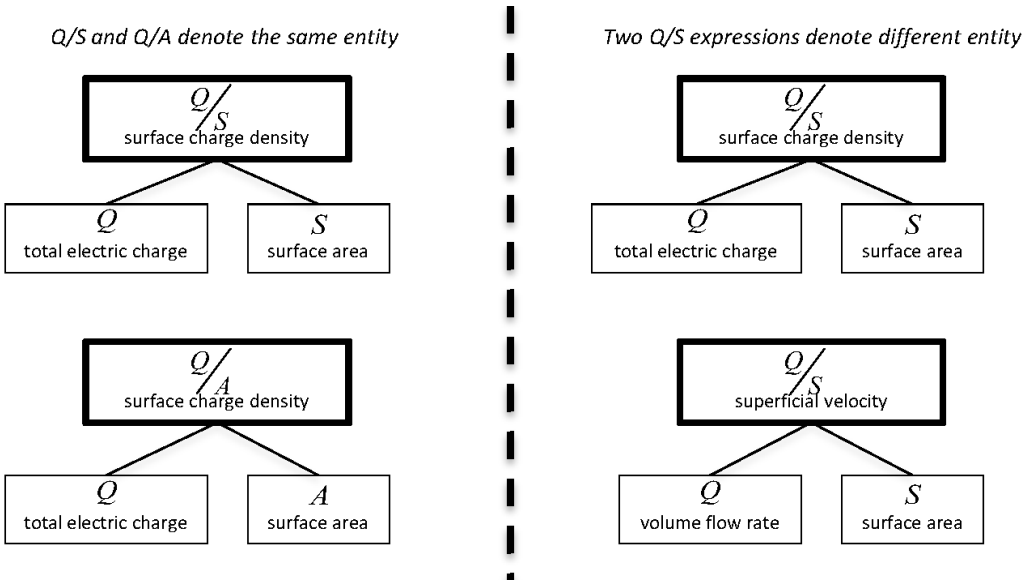


Figure 1.2: Dependency between the meaning of a math expression and the meanings of its constituent symbols. The left-hand part of the figure shows two math expressions that have different representations, yet express the same math concept. The right-hand part shows two math expressions that have the same representation, but different meanings.

of mathematical search system, (2) utilizing dependency relationships between mathematical expressions in mathematical search system, and (3) mathematical entity linking.

1.3.1 Framework of Mathematical Search System

The main objective in the research of MIR is to have a math search system capable of precisely retrieving math expressions relevant to users' queries. There are multiple aspects of a math search system that we need to consider during the development process:

- Extraction of textual information of each math expression in the document collection.

  The most popular way to extract text related to a math expression is simply by using all words found in the document or paragraph containing the expression. However, this approach also introduced an issue. Since all math expressions within the same paragraph or documents will have the same textual information, it is difficult to identify the exact meaning of each expression. To tackle this issue, we need to extract textual information that is more precise in defining math expressions.

- Indexing of math expressions and their textual information.

  Although textual information can be easily indexed using a current search

engine library, such as Lucene (The Apache Software Foundation, 2015a), math expressions require a preprocessing step prior to indexing. We need to extract the literals (identifiers, constants, and operators) and the substructures of math expressions. We also have to consider the placement of symbols in math expressions. All this extracted information about the content and structure of each math expression is then indexed. This is expected to allow a flexible search, thus we can obtain retrieval results with good recall.

- Scoring function for the ranking purpose.
  A reliable scoring function is required to allow high-precision retrieval. Since we may index several different types of information in our database, we need to derive a scoring function that prevents a bias toward certain types of information. Furthermore, we can incorporate a reranking technique as a post-processing tool.

Chapter 3 is dedicated to introducing our math search framework and describe in detail how we set up each aspect of the search module.

1.3.2 Utilizing Dependency Relationships between Mathematical Expressions in Mathematical Search System

Textual information explaining a mathematical expression is usually obtained from the text surrounding the expression. However, this surrounding text does not always contain the explanation for the symbols that exist within the expression. This issue can be addressed by naïvely setting the surrounding text to have a very wide window size. However, this risks including many words from the surrounding text that do not necessarily explain the expression or the constituent symbols. Therefore, instead of naïvely expanding the window size of the surrounding text, we propose to capture the dependency relationships between each math expression and its constituent sub-expressions in a document. For instance, given three math expressions $\frac{Q}{A}$, Q, and A, we extract two dependency relationships, i.e., between $\frac{Q}{A}$ and Q, and between $\frac{Q}{A}$ and A, as shown in Figure 1.2. Subsequently, we can use the meanings (as captured in the surrounding text) of Q and A to determine the meaning of the larger expression $\frac{Q}{A}$. In this dissertation, we use the term "dependency graph" to refer to the set of all constituent expressions and the set of ordered pairs showing individual dependencies.

In this dissertation, we extract the dependency graphs of math expressions by examining only the presentation of each math expression. Since each doc-

ument will have its own dependency graph, three characteristics of math expressions (Kohlhase and Sucan, 2006) that cause ambiguity will not be critical issues. Within a document, math expressions with the same visually rendered form often have the same meaning. However, the opposite does not always hold. Within a document, a mathematical concept is sometimes expressed by a notation with certain variations. For instance, the function $f(x)$ may be defined as $f(x) = x + c$ and $f(x) = x + 5$ in a document. These expressions should be related to one another, because both defined $f(x)$.

In this dissertation, we propose a heuristic method for constructing dependency graphs. We base our heuristic method on string matching between math expressions, with five normalization steps applied to each expression prior to the matching procedure. Our first objective in this task will be to measure how accurate the proposed method is in extracting the dependency graphs. Our hypothesis on this respect can be stated as:

Hypothesis 1 *A combination of normalizing math expressions and applying string matching between expressions can predict the existence of dependency relationships between math expressions.*

Further, this proposed dependency graph will allow us to enrich the textual information of each indexed math expression without the need to expand the window size of the surrounding text. Our second objective will be to evaluate the effectiveness of the enriched textual information to improve the performance of the math search systems. By doing so, we hypothesize that:

Hypothesis 2 *The use of a dependency graph helps determine the meaning of math expressions, thus improving the search results of math search systems.*

We test the Hypotheses 1 and 2 in Chapter **??** and **??**, respectively.

1.3.3 Linking Mathematical Expressions to Wikipedia

One of the challenges in developing an MEL system is shared with math search systems, that is, the semantics of math expressions are often difficult to identify from their surface level representation. Therefore, a solution that is effective for math search systems would also be expected to work for an MEL system. However, there is a unique challenge that appears in MEL, but not in math searches: math expressions in the Wikipedia articles that match a given math mention are likely to be important in the containing Wikipedia articles. This challenge arises because the nature of the task performed by MEL is to assign the Wikipedia title

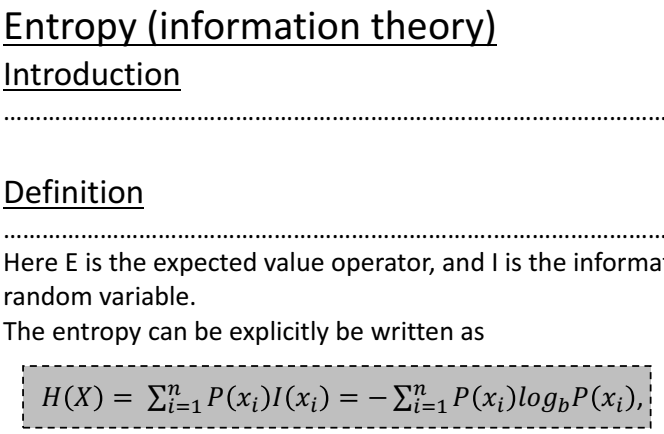


Figure 1.3: An example of a representative math expression in a Wikipedia article. There are two distinct math expressions, $H(X) = \sum_{i=1}^{n} P(x_i)I(x_i) = -\sum_{i=1}^{n} log_b P(x_i)$ and e. The former expression is more significant to the article than the latter.

to mentions. Hence, we expect that, given a math mention, the MEL system will return a Wikipedia article that contains math expression(s) similar to the mention, and more importantly, of which the title precisely describes the math mention.

We first investigate the performance of MEL by simply using the math search technique in Chapter 5.6. To provide more accurate links, we attempt to capture the importance of matching math expressions in their corresponding Wikipedia articles. In a Wikipedia article, early sections, such as the summary, infobox, introduction, and definition, hold explanations that are important to the concept described by the article. Therefore, math expressions that appear in these early sections are very likely significant to the article. In addition, each math expression in a Wikipedia article may be displayed either inline or in an equation-type environment (i.e., it appears on its own line). Important math expressions are often displayed in equation-type environments, and less significant expressions are displayed inline. Figure 1.3 illustrates an example of a representative/important math expression $H(X) = \sum_{i=1}^{n} P(x_i)I(x_i) = -\sum_{i=1}^{n} log_b P(x_i)$ in the Wikipedia article entitled "Entropy (information theory)." This math expression is found on the early section (i.e. "Definition") and displayed in an equation-type environment.

Our objective is to determine the best non-null Wikipedia article for each math expression given the observation of common features, such as math and text similarities, as well as the location and display of matching expression in the

article. Hence, our hypothesis can be stated as:

Hypothesis 3 *A combination of math search features (math and text similarities) and the importance feature, which is encoded from the location and display of the matching math expression in the Wikipedia article, can predict a correct link for each math mention.*

1.4 Contribution

1.4.1 Framework of Mathematical Search System

We develop the framework of our math search module by introducing three granularity levels of textual information (math, paragraph, and document) and combining them by first applying score normalization. Furthermore, we investigate the impact of using cold-start weights obtained from multiple linear regression. These methods will allow us to index multiple types of information in our databases, and prevent a bias toward certain types of information during scoring. In addition, we apply unification as a post-processing tool in our search module. The motivation is to ensure that math expressions that can be instantiated from the query are ranked higher than those that cannot.

We evaluate our math search module in the NTCIR-12 MathIR task (Zanibbi et al., 2016), and show a precision of 23.45% in relevant judgment and 48.28% in partially relevant judgment. In this task, our proposed search system shows the highest search performance among four other available systems.

1.4.2 Utilizing Dependency Relationships between Mathematical Expressions in Mathematical Search System

We next propose the concept of dependency graph between math expressions to further improve the performance of our math search module. First, we introduce a heuristic method to extract dependency graphs and evaluate its effectiveness using manually annotated data. Our evaluation shows that the extraction method delivers an accuracy of 86.74%.

Then, we validate the effectiveness of the dependency graph in enriching textual information related to math expressions and in improving the retrieval results of math search module. Our experimental result shows that the use of the dependency graph in the math search module increased the precision by 12.60%. Furthermore, in the NTCIR-12 MathIR task, we record that the dependency graph significantly improves the precision of our search module, i.e., a relative improvement of up to 24.52%.

1.4.3 Linking Mathematical Expressions to Wikipedia

The MEL module of our MIA system was developed by introducing a learning-based approach that exploits our proposed dependency graph and utilizes several features, such as math-level similarity, document-level similarity, math importance, and matching location. We encode the importance feature by applying the Personalized PageRank algorithm to the dependency graph of the math expressions that appear in each Wikipedia article. We utilize two factors to establish the preference vector, namely location and display of each math expression in the article. We found that this proposed importance feature is capable of detecting representative/important math expressions in Wikipedia articles.

In addition, to train and evaluate our approach, we construct a dataset that is derived from a dataset released by the NTCIR-12 MathIR Wikipedia task (Zanibbi et al., 2016). Our evaluation shows that the proposed approach achieves 83.40% precision, outperforming the straightforward application of a math search system (6.22%).

1.5 Dissertation Outline

The remainder of this dissertation is organized as follows.

- **Chapter 2 — Literature Review**
  We introduce several previous reports of work related to this dissertation. This work can be classified into three categories: development of math search systems, extraction of textual information related to math expressions, and the entity-linking task.

- **Chapter 3 — Framework of Mathematical Search System**
  In this chapter, we present the framework of our math search system in detail. We explain the techniques that were used to encode math expressions and the extraction of textual explanations for each math expression. Subsequently, we present our approach in combining all the indexed information for the searching purpose. Next, we describe the unification method as a post-processing module. Then, we report the performance of our math search system in the NTCIR-12 MathIR task.

- **Chapter 4 — Utilizing Dependency Relationships between Mathematical Expressions in a Mathematical Search System**
  We discuss the extraction and impact of the dependency graph of mathematical expressions in our math search system. We first explain the depen-

dency graph of mathematical expressions. Then, we describe our proposed heuristic extraction method and the application of the dependency graph in math search systems. We evaluate the effectiveness of our heuristic method in extracting dependency graphs of math expressions, and also examine the impact of these dependency graphs on our math search system.

- **Chapter 5 — Linking Mathematical Expressions to Wikipedia**
  We present our solution to the challenge of determining the identity of math expressions in documents by linking these expressions to their corresponding Wikipedia articles. We introduce a learning-based approach using common features, such as math and text similarities, as well as the importance of math expressions within the document. Further, we test our approach using a dataset that is derived from a dataset released by NTCIR-12 MathIR Wikipedia task.

- **Chapter 6 — Discussion and Future Work**
  In this chapter, we discuss the limitations of our math search and math entity linking modules. In addition, we present several future tasks and potential directions in which to continue our investigation.

- **Chapter 7 — Conclusion**
  Here we summarize our findings and their implications for the communities of mathematical information access, information retrieval, and entity linking.

Chapter 2

Literature Review

This chapter discusses previous work related to this dissertation which can be summarized into four categories: formats of math expressions, development of math search systems (including formula indexing, ranking function, and test collections), textual information related to math expressions, and the entity linking task.

2.1 Formats of Mathematical Expressions

There are several different formats for math expressions, including LaTeX, ASCIIMath, OpenMath, and MathML. LaTeX, ASCIIMath, and Presentation markup of MathML are used to display math expressions. On the other hand, OpenMath, and Content markup of MathML are used to convey mathematical meaning. Figure 2.1 shows a math expression expressed in several different formats.

2.1.1 LaTeX

TeX/LaTeX (Knuth, 1984; Lamport, 1994) is a typesetting markup language that are popular within scientific community. One of the greatest strengths of TeX(and later on, LaTeX) is that it allows simple construction of math expressions. As shown by the example in Fig. 2.1 , the markup codes in TeX/LaTeX are represented by strings beginning with a backslash, e.g., \frac and \sqrt.

2.1.2 MathML

MathML (Ausbrooks et al., 2014) is a markup language for representing math expressions. This format is recommended by the W3C Math Working Group as a standard to describe math expressions. It is usually expressed in XML syntax. An important aspect of MathML is that there are two types of markup: Presentation markup and Content markup.

LaTeX

```
\frac{\sqrt{a}}{2}
```

ASCIIMath

```
sqrt(a) / 2
```

Presentation MathML

```
<math>
  <mfrac>
    <msqrt>
      <mi>a</mi>
    </msqrt>
    <mn>2</mn>
  </mfrac>
</math>
```

Content MathML

```
<math>
  <apply>
    <divide/>
    <apply>
      <root/>
      <ci>a</ci>
    </apply>
    <cn>2</cn>
  </apply>
</math>
```

OpenMath

```
<OMOBJ>
  <OMBIND>
    <OMA>
      <OMS cd="arith1" name="divide" />
      <OMA>
        <OMS cd="arith1" name="root" />
        <OMV name="a" />
      </OMA>
      <OMI>2</OMI>
    </OMA>
  </OMBIND>
</OMOBJ>
```

Figure 2.1: Several different formats for representing a math expression $\frac{\sqrt{a}}{2}$

Presentation MathML

Presentation markup of MathML is used to describe the layout structure of mathematical notation. Compared to LaTeX, this markup contains additional tags to identify symbols types. The presentation elements of Presentation MathML are divided into two classes, namely token elements and layout schemata. Token elements are intended to represent the smallest units of math notation which carry meaning, e.g., individual symbols, function names, numbers, and labels. Meanwhile, layout schemata build expressions out of parts and can have only elements as content. These are further divided into General Layout, Script and Limit, Tabular Math, and Elementary Math schemata. All these MathML presentation elements only suggest specific ways of rendering; a particular MathML renderer is free to use its own rules.

Content MathML

Content MathML provides an explicit encoding of the underlying mathematical meaning of a math expression instead of its layout. The basic building blocks of Content MathML expressions are numbers, identifiers, and symbols. This markup uses the `<apply>` element to represent a function application.

2.1.3 ASCIIMath

MathML format is the best-known open markup format for representing math expressions, because it has a mature specification, and also supports two types of markup for different purposes. However, it may be inconvenient for people to write math expressions using MathML format, because it needs to be expressed in XML (or HTML) syntax.

ASCIIMath (Gray, 2007) overcomes this problem by providing an easy way to produce MathML representation. People write math expressions using ACIIMath markup, which is a simplified markup language, and then ASCIIMathML script converts this ASCIIMath notation to MathML.

2.1.4 OpenMath

OpenMath (Buswell et al., 2004) has a strong relationship to MathML Content. Both formats represent the semantics of math expressions. As shown in Fig. 2.1, there is an obvious line-by-line similarity for the XML structures and token elements. The main difference is the use of content dictionaries (`cd` attributes in OpenMath elements) to handle symbols. In the example, the identification of symbols `divide` and `root` relies on the availability of content dictionary `arith1`.

2.2 Development of Mathematical Search Systems

In the mathematical information retrieval (MIR), the user information needs can be classified into several categories as shown in Table 2.1. To satisfy these information needs, majority of current MIR research focus on the development of search techniques based on query-by-expression (Kohlhase and Kohlhase, 2007; Zhao et al., 2008; Zanibbi and Blostein, 2012).

2.2.1 Formula Indexing

Kamali and Tompa (2013a) found that searching for mathematical expressions and retrieving relevant documents based on their mathematical content is not straightforward. The following reasons were identified.

Table 2.1: Categories of information needs in math information retrieval (Kohlhase and Kohlhase, 2007; Zhao et al., 2008; Zanibbi and Blostein, 2012)

No	Information Need
1	Specific or similar math formulae (visual form/appearance, mathematical content, or concept name)
2	Theorems, proofs, and counter-examples
3	Examples and visualizations (e.g. graphs or charts)
4	Problems and solution sets (e.g. for instruction)
5	Algorithms
6	Applications (e.g. applications of Fourier transform)
7	Answer for mathematical questions or conjectures
8	People (e.g. authors)
9	Determine novelty/sequence of mathematical discoveries

- Mathematical expressions are objects with a complex structure and few distinct symbols and terms. Furthermore, the symbols and terms alone are usually inadequate for distinguishing between different math expressions.
- Relevant math expressions may include small variations in presentation.
- The majority of published math expressions are encoded with respect to their presentation, and most instances do not preserve sufficient semantic information.

As a result, formula indexing and retrieval has become one of the key issues in math retrieval (Zanibbi and Blostein, 2012). Many math search systems have attempted to solve this issue, with the most popular implementation reducing the search for formulae to a full-text search. There are several other techniques (Guidi and Sacerdoti Coen, 2015), such as substitution-tree indexing, reduction to SQL queries, and reduction to XML-based searches, but fewer systems have been derived from these. We now provide a review of several math search systems.

Reduction to Full Text Searches

The most popular implementation of a math search system is to reduce the search for formulae to a full-text search. Several early studies on math search systems, e.g. ActiveMath (Libbrecht and Melis, 2006), MathGO! (Adeel et al., 2008), DLMF (Miller and Youssef, 2003; Youssef, 2005, 2006, 2007a,b), MathFind (Munavalli and Miner, 2006), and Mathdex (Miner and Munavalli, 2007), implemented full-text search technology. In ActiveMath (Libbrecht and Melis,

2006), mathematical data are represented in OMDoc format. These data are tokenized and then indexed using the Lucene search engine. MathGO! (Adeel et al., 2008) uses a template-based approach to recognize and search for math expressions. MathFind (Munavalli and Miner, 2006) translates each math expression in a document into a sequence of text-encoded math fragments, and then indexes this sequence together with all words found within the document.

The Digital Library of Mathematical Functions (DLMF) search (Miller and Youssef, 2003; Youssef, 2005, 2006, 2007a,b) is a fully math-aware fine-grained search system that supports search and access to fine-grained targets such as equations, figures, tables, definitions, and named rules/theorems. In its implementation, this system handles mathematical expressions as a collection of mathematical terms. Hashimoto et al. (2007) and Hijikata et al. (2007) proposed a search engine for MathML objects using the structure of mathematical expressions, whereby inverted indices were constructed using the DOM structure and represented in XPath.

Several of the more recent math search systems can be classified as follows.

- *Extracting literals (identifiers, constants, and operators) from mathematical expressions.*

  The Qualibeta system (Pinto et al., 2014) extracts features such as categories, a set of identifiers, a set of constants, operators, and a set of unique identifiers to represent each math expression.

- *Extracting and generalizing the substructures of math expressions.*

  WikiMirs (Hu et al., 2013) is a tree-based indexing system that indexes all substructures of the formulae in LaTeX considering the generalization of substructures. Subsequent versions of WikiMirs (Lin et al., 2014; Gao et al., 2014; Wang et al., 2015) utilize a semantic enrichment technique to extract useful semantic information from math expressions, and then apply hierarchical generalization to the substructures of the expressions to support substructure matching and fuzzy matching. The MIaS system (Líška, 2013; Sojka and Líška, 2011b; Sojka, 2012; Růžička et al., 2014) focuses on a similarity search based on canonical MathML. It processes each math expression by applying ordering, subexpression extraction, variable unification, and constant unification steps to the math expression. TUW (Lipani et al., 2014) implements a tokenizer that starts from the tree structure of the math expressions, and then extracts and linearizes the tokens. This

tokenizer slices a math expression tree by levels, and collapses each subexpression obtained from the slicing step.

- *Extracting placement of symbols in math expressions.*
  The Tangent system (Pattaniyil and Zanibbi, 2014) captures the structure of math expressions by generating symbol pair tuples from a symbol layout tree representation of the expression. These symbol pair tuples describe the relative placement of symbols in an expression. The MCAT system (Topić et al., 2013; Kristianto et al., 2014c,a), captures the content and structure of math expressions by encoding the paths between nodes within the tree structure of the expressions.

Other Approaches

Another type of approach used in math search systems is the one based on substitution-tree indexing (Graf, 1996), which was originally developed for automatic theorem proving to store lemmas and quickly retrieve formulae up to instantiation or generalization. MathWebSearch (Kohlhase and Sucan, 2006; Kohlhase et al., 2012; Hambasan et al., 2014) is based on this approach. In addition, Schellenberg (2011); Schellenberg et al. (2012) introduced a system for the layout-based (LaTeX) indexing and retrieval of math expressions using substitution trees.

Other math search systems aim to approximate math expressions via a series of relations to be stored in a relational database. Asperti and Selmi (2004) generated an SQL query for each given query formula that is represented by a set of relations. The recall of their system can be maximized by relaxing such relations or by employing normalization. The fourth type of math search system reduces the search for math expressions to an XML-based search. For instance, the FSE system (Schubotz et al., 2013) formalized formula queries using XQuery/XPath technology. This system traded flexibility for performance, and did not normalize the input in any way. In addition to the systems described above, there is a lattice-based system developed by Nguyen et al. (2012b) that cannot be classified into any of the above categories. This system extracted math features from the MathML-formatted math expressions, and then constructed a mathematical concept lattice using these features.

2.2.2 Ranking Function for Formula Retrieval

In addition to math expression indexing, several studies have focused on determining the ranking function for formula retrieval. Yokoi and Aizawa (2009) proposed a similarity search scheme for mathematical expressions based on a Subpath Set and reported that it works well on "simplified" Content MathML. Sojka and Líška (2011a) computed a weight for each indexed math expression that describes how far this expression from its original representation. They then tried to create a complex and robust weighting function that would be appropriate to documents from many scientific fields. Nguyen et al. (2012a) proposed an online learning-to-rank model to learn scoring function after having extracted math expression features based on the MathML Content format. Their model outperformed other standard information retrieval models, but there was no comparison against other math-specific similarity functions.

Kamali and Tompa (2013a) proposed a tree-edit distance-based method to calculate the structural similarity between two expressions, and also introduced a pattern-based search to enable flexible matching of expressions in a controlled way. Furthermore, Kamali and Tompa (2013b) described optimization techniques to reduce the index size and the query processing time required by this tree-edit distance-based method. Zhang and Youssef (2014) proposed five math similarity measure factors: taxonomy of functions and operators, data-type hierarchical level of the math expressions, depth of matching position, query coverage, and the different importance between expression and formula. Schubotz et al. (2014) evaluated each of these five factors individually and confirmed that each factor is relevant to math similarity, with a note that the last factor mentioned above is of lower relevance than the other four factors.

2.2.3 Test Collections for Math Search Systems

Most early work on math search systems used specially generated datasets to evaluate the resulting systems. The number of publicly available test collections, which consist of a document set, list of topics, and assessment of pooled results, for the evaluation of math search systems is limited.

Kamali and Tompa (2013a)'s Dataset

Kamali and Tompa (2013a) generated an evaluation dataset that consists of pages from Wikipedia and DLMF, and contains a total of 863,358 math expressions. The 98 queries in this dataset were produced from an interview process (45)

and a mathematics forum (53). In the interview process, the invited students and researchers were asked to search for math expressions of potential interest to them in a practical situation. To prepare queries from mathematics forum, Kamali and Tompa (2013a) gathered discussion threads each of which can be described with a query that consists of a single math expression. They manually read each thread and responses to manually judge if a given math expression, together with the page that contains it, can answer the information need of the user who started the thread. Queries in their dataset contain math expressions, but no textual keywords.

The NTCIR-10 Math Pilot Task

The NTCIR-10 Math Pilot task (Aizawa et al., 2013) was the first attempt to develop a common evaluation dataset for math formula searches based on a pooling method. This dataset includes 100,000 scientific papers, 21 topics for formula searches (each query contains only math formulae), and 15 topics for full-text searches (each query contains both a list of formulae and a list of textual keywords). For each topic, 100 retrieval units (math expressions) from the pooled results were assessed. Topics in this task express several user information needs, such as specific or similar math formulae (category 1 in Table 2.1), theorems (2), examples (3), solutions (4), and applications (6).

The NTCIR-11 Math-2 Task

The NTCIR-11 Math-2 task (Aizawa et al., 2014) dataset consists of 105,120 scientific articles (with about 60 million math expressions) converted from LaTeX to an HTML+MathML-based format by the KWARC project (Kohlhase and Ginev, 2016). To get a varied sample of technical documents containing math expressions, this corpus contains articles from the arXiv categories math, cs, physics:math-ph, stat, physics:hep-th, and physics:nlin. In addition, this dataset contains 50 full-text search topics, which express the same information need categories as in the Math Pilot task. This task is focused on full-text searches, and uses paragraph-level retrieval units (8,301,578 units in the dataset). Furthermore, 50 retrieval units (paragraphs) from pooled results (from 20 submitted runs) have been assessed for each topic. Each retrieval unit is judged as being non-relevant, partially relevant, or highly relevant. In addition to this Math-2 task, the Wikipedia Open Subtask (Aizawa et al., 2014; Schubotz et al., 2015) used the Wikipedia dataset instead of scientific articles.

The NTCIR-12 MathIR Task

The NTCIR-12 MathIR task (Zanibbi et al., 2016) makes use of two corpora. The first corpus contains technical articles in asXiv, and the second corpus contains English Wikipedia articles. For each corpus, there are two subtasks. Three subtasks (arXiv-main, arXiv-simto, and Wiki-main) contain queries with formulae and keywords. The last subtask (Wiki-formula) considers isolated formula queries. The arXiv corpus used in this task is the same as the one used for NTCIR-11 Math-2. The retrieval units (paragraphs) for this corpus is also the same one used for NTCIR-11 Math-2.

The Wikipedia corpus provided by this task contains 319,689 articles from English Wikipedia converted into a simpler XHTML format with images removed. Unlike the arXiv corpus, the retrieval units for this corpus is document. Around 10% of the sampled articles contain `<math>` tags that demarcate LaTeX. All articles with a `<math>` tags are included in the corpus. The remaining 90% of the articles are sample from Wikipedia articles that do not contain a `<math>` tag. The task organizer set this latter set of articles as distractors for keyword matching.

The Wikipedia articles were initially in MediaWiki format. The NTCIR organizers extracted the math expressions by first converting the MediaWiki math templates to LaTeXand then converted them together with LaTeXformulae demarcated by `<math>` tags in the articles to MathML format using LaTeXML (Miller, 2016). In total, there are 592,442 formulae in the corpus, encoded using LaTeX, Presentation MathML Presentation, and Content MathML. There was however no attempt from the task organizer to detect or label untagged formulae, which often appear directly in HTML text.

MathOverflow Dataset

The general-purpose Mathematical Information Retrieval (MIR) test collections mentioned above are focused on accommodating information needs of varying complexity. A recent work by Stathopoulos and Teufel (2015) focused on the retrieval of research-level mathematical information and the construction of a Math IR test collection for these needs. They regarded the questions in MathOverflow as the information needs, and then tried to retrieve the scientific publications that answer these questions. To construct the topics, they examined each identified MathOverflow discussion thread for conformance to two criteria: (a) useful questions should express an information need that is clear and can be satisfied

by describing objects or properties, stating conditions, and/or producing examples or counter-examples and (b) scientific documents cited in the MathOverflow accepted answers should address all sub-parts of the question in a manner that requires minimal deduction and do not synthesize mathematical results form multiple resources. Topics in their dataset take the form of coherent text interspersed with math expressions. The relevance judgments for these topics were procured from the answers of the corresponding discussion threads in the MathOverflow.

2.3 Extraction of Textual Information for Mathematical Expressions

Previous work on extracting textual information for math expressions can be classified into two categories. The first assumes that the meaning of a math expression can be found from text preceding and/or following the expression (Grigore et al., 2009; Wolska et al., 2011; Pinto and Balke, 2015), and the second attempts to extract only those terms that precisely describe the math expression (Nghiem et al., 2010; Yokoi et al., 2011; Kristianto et al., 2014b).

2.3.1 Utilization of Surrounding Text to Capture the Meaning of Math Expressions

One of the earliest studies on extracting textual information for mathematical expressions actually focused on resolving the semantics of mathematical expressions (Grigore et al., 2009). This approach uses the natural language within which math expressions are embedded to resolve their semantics and enable their disambiguation. It focuses on mathematical expressions that are syntactically part of a nominal group and in an apposition relation with the immediately preceding noun phrase, i.e. the target expressions come from a linguistic pattern: "... *noun_phrase symbolic_math_expression*...". By assuming that a target mathematical expression can be disambiguated using its left context, this method disambiguates mathematical expressions by computing the term similarity between the local lexical context of a given expression, i.e. all nouns appearing in the five-word window to the left of the target expression, and a set of terms from Term Clusters based on the OpenMath Content Dictionaries[1]. The calculated similarity score determines the disambiguation of the target expression. Wolska and Grigore (2010) complemented the work of Grigore et al. (2009) by conducting three corpus-based studies on declarations of simple symbols in mathematical

[1] `http://www.openmath.org/cdnames.html`

writing. This work counted how many mathematical symbols were explicitly declared in 50 documents. Subsequently, Wolska et al. (2011) determined that each target mathematical expression had a local and global lexical context. The local context is a set of domain terms that occur within the immediately preceding and following linguistic context. The global context is a set of domain terms that occur in the declaration statements of the target expression or other expressions that are structurally similar to the target expression. These two types of lexical context were then utilized to enhance the disambiguation work. Another recent study by Pinto and Balke (2015) extracted the sense of math expressions by applying Latent Dirichlet Allocation over context words surrounding the expressions. They concluded that the senses obtained from the context words were helpful for classification, such as predicting the Mathematics Subject Classification (MSC) classes from a document collection.

2.3.2 Extracting Specific Descriptions of Math Expressions

Nghiem et al. (2010) proposed text matching and pattern matching methods to mine mathematical knowledge from Wikipedia. These methods extracted coreference relations between formulae and the concepts that refer to them. Yokoi et al. (2011) extracted textual descriptions of mathematical expressions from Japanese scientific papers. This work considered only compound nouns as the description candidates. When the descriptions are expressed as complex noun phrases that contain prepositions, adverbs, or other noun phrases, only the final compound noun in the phrase (Japanese is a head-final language) is annotated and extracted. A subsequent study (Kristianto et al., 2012a) proposed a set of annotation guidelines by introducing full and short descriptions, and enabling multiple descriptions to be related. Following this, two applications based on the extracted descriptions of mathematical formulae were introduced, namely semantic search and semantic browsing (Kristianto et al., 2012b).

2.3.3 Utilization of Textual Information in Math Search Systems

Prior to the indexing process, several math search systems extract textual information to represent mathematical expressions. These systems store both the math expressions and the associated textual information in their index systems to enable mathematical expressions to be searched using queries that contain both math expressions and textual keywords. Nguyen et al. (2012a) presented a math-aware search engine that handles both textual keywords and mathematical expressions, and benchmarked the system using math documents crawled from an

online math question answering system, MathOverflow[2]. Several systems (Hambasan et al., 2014; Růžička et al., 2014, 2016; Pattaniyil and Zanibbi, 2014; Davila et al., 2016; Lipani et al., 2014) construct word vectors to represent each math expression by associating the expression with all words found in the same document or paragraph. WikiMirs 3.0 (Wang et al., 2015; Gao et al., 2016) regards the preceding and following paragraphs of a math expression as the context of the expression. Similarly, Qualibeta (Pinto et al., 2014) and MCAT (Kristianto et al., 2014c) associate each expression with the surrounding text. We consider that, when surrounding text is used to represent math expressions, the proportion of words related to that expression is higher than when all words in the document are used.

2.4 Wikification

In natural language processing, entity linking (EL) is a task to overcome the ambiguity and variability of natural language. Mentions of entities may have multiple meanings and a given entity can be expressed in many ways. EL addresses these challenges by linking entity mentions in text to corresponding entities in a knowledge base. The definition of this task consists of several aspects as follows.

- Definition of the mentions to highlight (e.g. named entities, keyphrases, and biological terms)
- Target encyclopedic resource (e.g. Wikipedia and ontologies)
- What to point in the knowledge base (e.g. Wikipedia title and concept in an ontology)

One variant of EL is wikification (Bunescu and Paşca, 2006; Cucerzan, 2007; Mihalcea and Csomai, 2007; Milne and Witten, 2008; Ferragina and Scaiella, 2010; Han et al., 2011; Ratinov et al., 2011; Cheng and Roth, 2013; Guo and Barbosa, 2014), which identifies a set of entity mentions in a document and then locates the most accurate mapping from these mentions to corresponding Wikipedia articles. Figure 2.2 shows an example of the wikification process. There are several subtasks to address in wikification (and in general, EL) as follows.

- *Identifying target mentions* in the input text that should be wikified.
- *Identifying candidate titles* that may correspond to each mention.
- *Disambiguating each mention* by ranking the candidate titles.

[2] `http://mathoverflow.net`

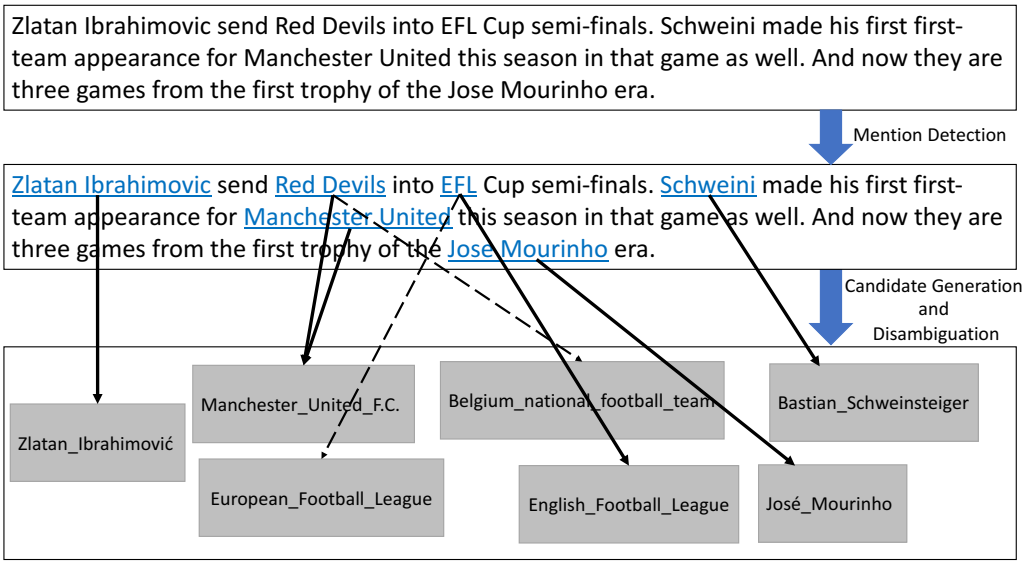


Figure 2.2: Processing steps in wikification. The first box contains the original text. The second box contains the same text, only the mentions are now detected (the underlined and blue colored text). The third box contains several candidate titles corresponding to the mentions. The arrows from the second to the third box illustrate the assignments of the candidate titles. The solid arrows indicate the assignment returned by the disambiguation step.

- *Detecting NIL mentions*, which are mentions that do not correspond to a Wikipedia title.

2.4.1 Mention Identification

For identifying target mentions, the simplest technique is to treat each n-gram as a potential concept mention. However, this approach is likely intractable for large document collections. An alternative to this is to apply surface form based filtering, e.g., extracting noun phrases (NPs) from shallow parsing, augmenting NPs with surrounding tokens, and removing stop words and punctuation from the extracted NPs. In addition to this, there are several studies applying classification and statistics based filtering techniques (Finkel et al., 2005; Ratinov and Roth, 2009; Li et al., 2012; Florian et al., 2006; Li and Ji, 2014; Mihalcea and Csomai, 2007; Mendes et al., 2011). Wikify! system (Mihalcea and Csomai, 2007) used this approach and assign a numeric value (i.e. tf-idf (Salton and Buckley, 1988), χ^2 independence test (Manning and Schütze, 1999), and keyphraseness measure) to each possible mention, reflecting the likelihood that a given candidate is a valuable keyphrases.

Mention detection modules often utilize a controlled vocabulary derived from prior link knowledge in Wikipedia. For instance, mentions can be defined as n-grams used as anchor text within Wikipedia (Ratinov et al., 2011; Davis et al.,

2012; G. Demartini and Cudré-Mauroux, 2012; Han and Sun, 2011; Han et al., 2011; Mihalcea and Csomai, 2007; Cucerzan, 2007; Milne and Witten, 2008; Ferragina and Scaiella, 2010). On the other hand, other systems only use terms that exceed link probability threshold (Bunescu and Paşca, 2006; Cucerzan, 2007; Fernandez et al., 2010; Hachey et al., 2013). DBPedia Spotlight (Mendes et al., 2011) uses dictionary-based chunking with string matching via DBPedia (structured content from Wikipedia) lexicalization dataset.

2.4.2 Generating Candidate Titles and Disambiguating Mentions

The candidate generation is usually based on matching each mention to the Wikipedia titles. For example, the candidates can be titles that are overlap, superstring, or substring of the mention. Then, the initial title ranks are based on Wikipedia article length, the number of incoming Wikipedia links, prior link probability (e.g. commonness (Medelyan et al., 2008)), or graph based methods (e.g. centrality (Hachey et al., 2011)).

Most of the EL and wikification studies focused on developing algorithms for disambiguation. Their proposed methods can be divided into: local and global disambiguation approaches.

Local Disambiguation

Early EL studies (Bunescu and Paşca, 2006; Mihalcea and Csomai, 2007) applied local methods, disambiguating each mention in isolation—usually ranking the candidate entities by their compatibilities with the mention—and then taking the most compatible entity. Let $M = \{m_1, \ldots, m_N\}$ be the set of mentions to disambiguate, and $\phi(m_i, t_j)$ be a score function representing the likelihood that candidate t_j is the correct disambiguation for mention m_i, the local approach returns assignments $\Gamma = (t_1, \ldots, t_N)$ by solving the following problem:

$$\Gamma^*_{local} = \arg\max_{\Gamma} \sum_i \phi(m_i, t_i) \tag{2.1}$$

Local disambiguation works well when the context is rich enough to identify a mention. Typical features (Dredze et al., 2010; Anastacio et al., 2011) used for ranking in this approach include popularity features, text-based document similarity, topical similarity (based on LDA (Blei et al., 2003)), name similarity, NER and Geo-based features, page-type features, and validation-only features.

Given all extracted features, wikification systems rerank their initial rankling lists. Several methods used for this reranking purpose can be categorized into: un-

supervised learning (Ferragina and Scaiella, 2010), supervised learning (Bunescu and Paşca, 2006; Mihalcea and Csomai, 2007; Milne and Witten, 2008; Lehmann et al., 2010; de Pablo-Sanchez et al., 2010; Han and Sun, 2011; Chen and Ji, 2011; Ratinov et al., 2011), and graph-based ranking (Gonzalez et al., 2012).

Global Disambiguation

More recent EL systems have adopted a global coherence approach that accounts for the semantic relations between mentions and entities, utilizing various measures of semantic relatedness. Figure 2.2 shows the motivation of this approach. Mention "Red Devils" have several different title candidates, e.g., `Manchester_United_F.C.` and `Belgium_national_football_team`. However, because the surrounding mentions are semantically closer (i.e. "'Zlatan Ibrahimovic", "Schweini", and "Jose Mourinho" are the Manchester United personnel) to the former entity than the latter one, this mention should be linked to `Manchester_United_F.C.`.

The global optimization problem to be solved by this approach is:

$$\Gamma^* = \arg\max_{\Gamma}[\sum_i \phi(m_i, t_i) + \psi(\Gamma)] \tag{2.2}$$

where ψ is a coherence function. As this formulation is NP-hard, many approximations have been introduced.

Cucerzan (2007) maximized the agreement between the contextual information extracted from Wikipedia and from a document, as well as the agreement among category tags associated with the candidate entities. Milne and Witten (2008) utilized directly connected entities to represent each entity and measured their relatedness using normalized Google distance (Cilibrasi and Vitanyi, 2007). Han et al. (2011) proposed a graph-based collective system exploiting the random walk model on an entity graph to jointly link mentions. Ratinov et al. (2011) introduced an EL system that combines local and global features. They analyzed the strengths and weaknesses of local and global variants of their EL system, GLOW, finding that local disambiguation provides a baseline that is difficult to surpass. Ceccarelli et al. (2013) investigated the effectiveness of combining a set of relatedness features together using a learning-to-rank technique. These features are shown in Table 2.2.

Cheng and Roth (2013) formalized the EL problem as an integer linear programming problem to find assignments collectively. Guo and Barbosa (2014) proposed the use of the probability distribution resulting from a random walk

Table 2.2: Relatedness Features (Ceccarelli et al., 2013)

Singleton Features	
$P(a) = \|\|in(a)\|\|/\|W\|$	probability of a mention to entity a
$H(a)$	entropy of a

Asymmetric Features	
$P(a\|b) = \|in(a) \cap in(b)\|/\|in(b)\|$	conditional probability of the entity a given b
$Link(a \rightarrow b)$	whether a link to b
$P(a \rightarrow b)$	probability that a links to b
$Friend(a, b)$	equals 1 if a links to b, and $\|out(a) \cap in(b)\|/\|out(a)\|$ otherwise
$KL(a\|\|b)$	Kullback-Leibler divergence

Symmetric Features	
$\rho^{MW}(a, b)$	co-citation based similarity
$J(a, b) = \frac{in(a) \cap in(b)}{in(a) \cup in(b)}$	Jaccard similarity
$P(a, b)$	joint probability of entities a and b
$Link(a \leftrightarrow b)$	whether a links to b and vice versa
$AvgFr(a, b)$	average of $Friend(a, b)$ and $Friend(b, a)$
$\rho^{MW}_{out}(a, b)$	ρ^{MW} considering outgoing links
$\rho^{MW}_{in-out}(a, b)$	ρ^{MW} considering the union of incoming and outgoing links
$J_{out}(a, b)$	Jaccard similarity considering the outgoing links
$J_{in-out}(a, b)$	Jaccard similarity considering the union of incoming and outgoing links
$\chi^2(a, b)$	χ^2 statistic considering incoming links
$\chi^2_{out}(a, b)$	χ^2 statistic considering outgoing links
$\chi^2_{in-out}(a, b)$	χ^2 statistic considering the union of incoming and outgoing links
$PMI(a, b)$	point-wise mutual information considering incoming links

with restart over a suitable entity graph as a unified representation of the semantics of entities and documents. Alhelbawy and Gaizauskas (2014) presents a collective disambiguation approach using a graph model.

2.4.3 NIL Detection

Most of the early studies in wikification used hyperlinks in the Wikipedia pages to train and evaluate their systems. They treated the anchor text in the hyperlinks as the mentions, and the Wikipedia articles the hyperlinks go to as the gold-standard data. During training, wikification systems used these mentions They treated these hyperlinks as disambiguated mentions.

This approach however, may introduce bias to the trained statistical models, because compared to the mentions from general text, mentions found in the Wikipedia are disproportionately likely to have corresponding Wikipedia articles.

According to Ratinov et al. (2011), a mention is linked to NIL entity if:

1. the mention does not have a corresponding Wikipedia article,
2. the mention has a corresponding Wikipedia title, but it does not appear the candidates, or
3. the disambiguation module chooses an incorrect disambiguation over the correct one.

Bunescu and Paşca (2006); Zuo et al. (2014) tackled this issue by returning the disambiguation result if its confidence score is greater than a threshold value, and NIL otherwise. Meanwhile, Ratinov et al. (2011) trained a classifier using several features, such as features from disambiguation step, disambiguation confidence, commonness entropy, and whether the mention is detected by named entity recognizer (NER) as a named entity, to decide whether the disambiguation result of a mention is indeed correct.

2.4.4 Recent Progress in Wikification

One of the interesting advances in wikification and entity linking is the implementation of Joint Entity Recognition and Linking (JERL). This approach attempts to improve the performance of end-to-end wikification by applying global inference using additional knowledge. The motivation of doing this approach is to transfer knowledge between entity recognition and linking modules. By doing so, the error propagation in EL or wikification systems with the pipeline architecture can be minimized.

Joint optimization is however expensive as it increases the complexity of the problem. It requires a careful consideration of features and mutual dependency between multiple tasks, so that the training and inference is tractable. Meij et al. (2012) proposed a solution to the problem of wikifying microblog posts. They studied several supervised machine learning methods, but without utilizing any global evidence. Similar to this work, Guo et al. (2013) also introduced a solution for microblog post (tweet) entity linking. They introduced a structural SVM algorithm (Chang et al., 2010; Taskar et al., 2004; Tscohantaridris et al., 2005), which requires an NP-hard inference, and utilized entity-to-entity relations, but did not incorporate evidence from multiple tweets. Huang et al. (2014) enhanced

these two previous studies by proposing a semi-supervised graph regularization model to incorporate both local and global evidence from multiple tweets.

Sil and Yates (2013) introduced a reranking model that exploits the dependency between entity linking decisions and mention boundary decisions. They exploited an existing state-of-the-art NER system to over generate candidate mentions, and left the linking algorithm, which is a linear MaxEnt based reranking model, to make a final decision. In contrast to the Guo et al. (2013)'s approach, Sil and Yates (2013)'s is more suited for long documents. While Sil and Yates (2013) depended on an existing NER system, Luo et al. (2015) captured the mutual dependency between NER and EL by considering entity type and confidence information. In this model, NER can also benefit from the decision of entity linking module, since both decisions are made together.

Chapter 3

Framework of Mathematical Search System

Current mathematical search systems are expected to allow math expressions within a document to be queried using mathematical expressions and keywords. Recent shared tasks in mathematical information retrieval (MIR) (Aizawa et al., 2013, 2014; Zanibbi et al., 2016) specified this requirement in their task definitions. However, among current math search systems, the majority of them have not yet fully exploited the text in the documents. Most of the previous work on math expression search focused on establishing index systems. They utilize only a few types of textual information. Hambasan et al. (2014); Růžička et al. (2016); Davila et al. (2016); Lipani et al. (2014) associate each math expression with all words found in the same document or paragraph. WikiMirs 3.0 (Wang et al., 2015; Gao et al., 2016) regards the preceding and following paragraphs of a math expression as the context of the expression.

To associate each math expression with all words found in the document or paragraph allows math search systems to have good recall performances. However, there is also negative impact introduced by this approach. Since all math expressions within the same paragraph or documents will have the same textual information, it is difficult to identify the exact meaning of each expression. Therefore, the use of such type of textual information is not effective in improving the precision of search system. We propose the use of textual information specifically describing the meaning of math expression to tackle this issue.

In this chapter, we take advantages of multiple types of textual information to build a reliable math search system. For each math expression, we store the textual information obtained from its containing paragraph and document, so that we can ensure that the recall performance of our system is good. Furthermore, we introduce our proposed textual information, which specifically defined each math expression, to our math search system. This will allow our system to have a good precision and recall.

Once we indexed math expressions and their textual information, we set up the scoring and ranking components of our system. We apply normalization to the scores from all information types, including structures of math expressions and textual information. The intent of this procedure is to prevent a scoring bias toward certain information types. Later, we explore several learning to rank methods to effectively combine all these information types we have in our database. Subsequently, we examine the effectiveness of unification technique as the final step in our math search pipeline. The results from recent MIR shared task (Zanibbi et al., 2016) show that our system outperforms four other math search systems (Gao et al., 2016; Růžička et al., 2016; Davila et al., 2016; Thanda et al., 2016).

3.1 Components of Mathematical Search System

Figure 3.1 shows the components of our math search system. We group them into three main building blocks:

1. Data processing prior to indexing
    - encode math expressions
    - extract textual information
2. Indexing math expressions
3. Searching and Ranking
    - combine all the indexed types of information
    - perform unification to the initial ranked list

The detail of each component is described in the next sections.

In the development of our math search system, we assume that math expressions are available in MathML format, since several tools can generate MathML from LaTeX (e.g. LaTeXML (Miller, 2016)), PDF files (e.g. Maxtract (Baker et al., 2012)), handwritten text (e.g. Infty Reader (InftyProject, 2015)), or digitized documents (e.g. Infty Reader).

3.2 Encoding Math Expressions

3.2.1 The Problem

Mathematical notation is quite inconsistent, and symbol set limited: a notation is commonly reused, and there often exist several different ways of writing down

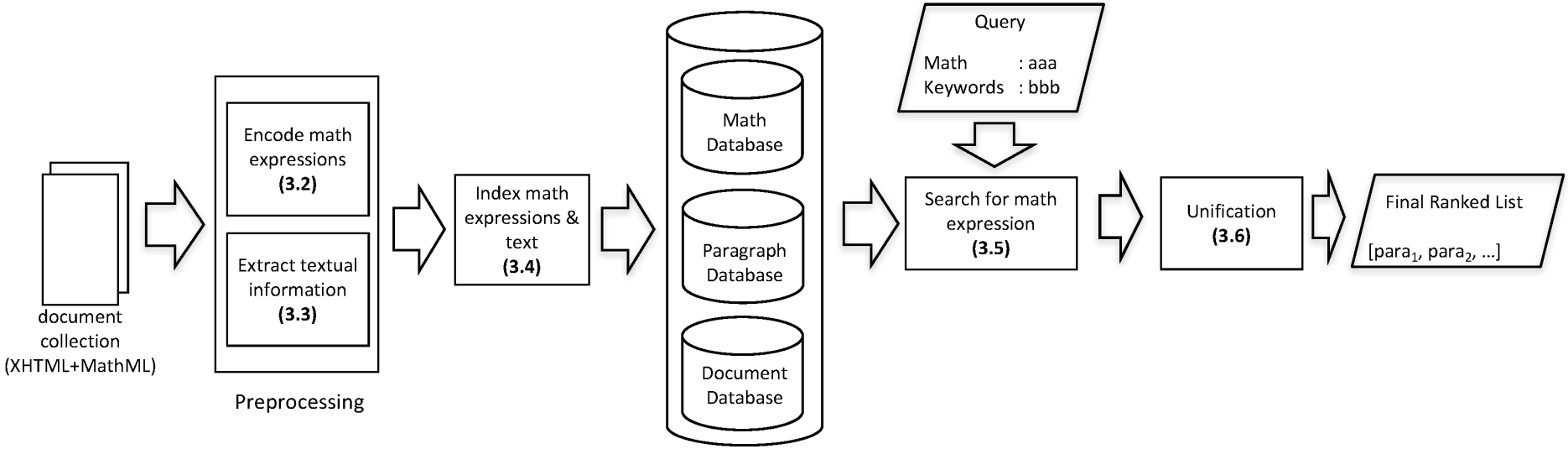


Figure 3.1: Overview of our math search system

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 $3\mathrm{i}+5$. 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 math expression 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.

The inability to reliably extract the correct semantic of an expression affected our method significantly. The 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 $\frac{x}{y}z$.

3.2.2 Our Method

The full-text search requirement was solved easily, by adopting Apache Solr (The Apache Software Foundation, 2015b), a popular open-source full-text search engine. Starting from this choice, we used the method described below to index and search the math expressions, and thus provide a flexible and soft-matching.

There are two techniques we employ to encode math expressions: path-based and hash-based techniques. Both techniques assume that math expressions are in the MathML format and treat each expression as a tree data structure.

Path-based Encoding Technique for Presentation MathML

The path-based encoding in Presentation MathML will capture the relative location of symbols (identifiers, operators, and literal number) in math expressions. Given a math expression in Presentation MathML format, this encoding technique produces five fields (types of information): `p_opaths_op`, `p_opaths_arg`, `p_upaths_op`, `p_upaths_arg` and `p_sisters` — which are described below in detail.

Each math expression, both at index time and at query time, is transformed into a sequence of keywords across several fields. First, vertical paths to operators (`<mo>`) are gathered into the `p_opaths_op` (ordered paths to operators) field in such a way that ordering is preserved. Meanwhile, vertical paths to identifiers (`mi`) and numbers (`mn`) are gathered into the `p_opaths_arg` (ordered paths to arguments) field. 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 `p_upaths_op` (unordered paths to operators) and `p_upaths_arg` (to arguments) with exactly the same information as in `p_opaths_op` and `p_opaths_arg` 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 fifth and final field carrying the expression structure, `p_sisters`, lists the sister nodes in each subtree.

```
<math>
  <mrow>
    <mi>x</mi>
    <mo>+</mo>
    <mi>y</mi>
  </mrow>
</math>
```

Figure 3.2: Content MathML example: $x + y$

Figure 3.2 presents a simple math expression, $x + y$, written in Presentation MathML. The `<mrow>` node groups the whole expression $x + y$. The `<mi>` and `<mo>` nodes represent an identifier and an operator, respectively. To index this mathematical expression, our search system encodes it into the form shown in Figure 3.3. If the visited element is a leaf (an operator `<mo>`, a numeric literal `<mn>`,

p_opaths_op: 2#mo#+
p_opaths_op: mo#+
p_opaths_arg: 1#mi#x 3#mi#y
p_opaths_arg: mi#x
p_opaths_arg: mi#y
p_upaths_op: #mo#+
p_upaths_op: mo#+
p_upaths_arg: #mi#x #mi#y
p_upaths_arg: mi#x
p_upaths_arg: mi#y
p_sisters: mi#x mo#+ mi#y

Figure 3.3: Encoding result of $x + y$ (Presentation MathML)

or an identifier <mi>), this encoding process lists not only the node, but also the value of the node. Given the root node (<mrow>), p_opaths_op tells that the second child is an operator element, which is <mo>, namely p_opaths_op: 2#mo#+. The p_upaths_arg lists the rest of the children (identifiers): p_opaths_arg: 1#mi#x 3#mi#y. The final elements visited by p_opaths_op and p_opaths_arg are the leaves. This is reflected by the encoding results: p_opaths_op: mo#+, p_opaths_arg: mi#x, and p_opaths_arg: mi#y.

The procedure for generating p_upaths is the same as the one for p_opaths, only with the ordering information removed. The upaths: #mo#+ suggests that a top-level element is an operator +. The final field, sisters, stores the sister nodes obtained from the only subtree in $x + y$ (mi#x mo#+ mi#y). Later, the matching process is performed on the MathML query, converting it into a Solr (The Apache Software Foundation, 2015b) disjunctive query.

This encoding technique is used at both index-time and query-time. While p_opaths and p_upaths are generated for every subtree of the math expression being indexed, at query time, however, p_opaths and p_upaths always start from the root of the math expression.

Path-based Encoding Technique for Content MathML

The motivation of path-based encoding for Content MathML is similar to to that for Presentation MathML. However, unlike paths extracted from Presentation MathML that capture only the relative positions of symbols in math expressions, each path captured from Content MathML also contains the relationships between operators and their arguments (identifiers-or-numbers). We suggest that

this will support our search system in detecting math expressions relevant to given queries.

Given a math expression in Content MathML format, this encoding technique produces four fields, namely `c_opaths_type`, `c_opaths_arg`, `c_upaths_type`, and `c_upaths_arg`. The general idea applied in this technique is the same as the previous one, namely to extract the paths in math structure trees. We make slight modifications here:

- In addition to the relative positions of symbols in math expressions, we extract the relationships between operators and their arguments (identifiers-or-numbers). To do so, for each given math subtree, we set the name of the function applied to all children as the name of the subtree root.
  Each subtree in a math expression in MathML Content is an `<apply>` subtree. In addition, each subtree contains exactly one operator (e.g. `<plus/>`, `<minus/>`, and `<divide/>`) as its child, and it is always the first child. Therefore, we can modify each subtree by replacing the name of its root with the name of its operator, then removing this operator from the children. If the operator is an element with a value (e.g. `<csymbol>-</csymbol>`), we concatenate its name and its value (`csymbol:-`), then set the concatenation result as the replacement for the root name. After this process, the resulting subtree still maintain the same semantic as the original.

- In addition to the vertical paths that express symbol names (`c_opaths_arg` and `c_upaths_arg`), we extract vertical paths expressing the symbol types (`c_opaths_type` and `c_upaths_type`).
  The idea is to allow an exact match by capturing the symbol name, and a partial match by capturing the symbol type. For instance, given a query $x + y$, we can now obtain $a + b$ in a higher rank than $1 + 2$, because both the query and $a + b$ represent an addition of two identifiers (x, y, a, and b are represented as `<ci>`). On the other hand, $1 + 2$ is an addition of two numbers (`<cn>`).

- We overlook `sisters` field for Content MathML, because it will store very similar information as `p_sisters`, only the tag names of the MathML elements are different.
  The `<mi>` and `<mn>` elements in Presentation MathML correspond to `<ci>` and `<cn>` in Content MathML, respectively.

Figure 3.4 presents $x + y$, written in Content MathML. First, we use the element name of `<plus/>` to replace `<apply>`. Next, we extract the vertical paths

```
<math>
  <apply>
    <plus/>
    <ci>x</ci>
    <ci>y</ci>
  </apply>
</math>
```

Figure 3.4: Content MathML example: $x + y$

c_opaths_type: `plus#1#ci`
c_opaths_type: `plus#2#ci`
c_opaths_type: `ci`
c_opaths_type: `cn`
c_opaths_arg: `plus#1#ci#x`
c_opaths_arg: `plus#2#ci#y`
c_opaths_arg: `ci#x`
c_opaths_arg: `ci#y`
c_upaths_type: `plus##ci`
c_upaths_type: `plus##ci`
c_upaths_type: `ci`
c_upaths_type: `cn`
c_upaths_arg: `plus##ci#x`
c_upaths_arg: `plus##ci#y`
c_upaths_arg: `ci#x`
c_upaths_arg: `ci#y`

Figure 3.5: Encoding result of $x + y$ (Content MathML)

from the new root `<plus/>` (in the place of the previous `<apply>`), to all children. The `c_opaths_type: plus#1#ci` describes that the first argument of operator `plus` is an identifier. In addition, `c_opaths_arg: plus#1#ci#x` describes that the first argument of this operator is variable x. The procedure for generating `c_upaths` fields is the same as the one for `c_opaths` fields, only with the ordering information removed.

The example above illustrates the operators-arguments relationships captured by path-based encoding from Content MathML. Such relationships can be captured from content markup, because this markup represents mathematical objects as expression trees. Content MathML includes information about each operation applied to a set of arguments. Therefore, we can extract the relationships of

operators and their arguments. This however, cannot be done in Presentation MathML, because presentation markup focuses on how a math expression is visually rendered, not on the underlying mathematical meaning of an expression.

Hash-based Encoding Technique

In addition to the path-based technique, we consider extracting subtrees from math expressions. To do so, we apply hash-based technique (Ohashi et al., 2016) to encode math expressions. There are three algorithms we use here, namely subtree hash, modular trick, and SIGURE hash, and their brief descriptions are as follows.

- *subtree hash* is designed to capture the implicit semantics represented by variable names. This algorithm is intuitively a depth-first pre-order traversal to construct hash codes, visiting each node once, and at each node having to combine (already computed) hash codes from children. The output from this algorithm is a feature set consisting of all the subtrees of the input tree.

- *modular trick* is designed to capture the semantics of structures or patterns in the formulae. For instance, the structure of $y = x^2$ indicates a quadratic equation, which in this example the left-hand side variable (y) is defined as the square of another variable (x). Based on this modular trick algorithm, both $y = x^2$ and $b = a^2$ have the same semantics. The output of this algorithm is a feature set consisting of all the substructures of a specified depth d rooted on any node of the tree.

- *SIGURE hash* is an algorithm to provide a metric which is invariant to variable names. In a math expression, alpha equivalent transformation corresponds to renaming of variables. For instance, this algorithm can capture the mathematician's intuition that $x = x$ has fundamentally a similar meaning to $y = y$, but not to $x = y$, for any value of x and y. The feature set obtained from this algorithm consists of subtrees of the input tree where all the variables are renamed according to their appearance order.

For an example, given math expression $\sum_{i=k}^{n} a_i$, whose Presentation MathML format is shown in Fig. 3.6, its hashing results are shown in Fig. 3.7. Each subtree from the subtree hashing is rooted at a certain node from the original tree and contains all of its descendants. Meanwhile, each subtree from the modular trick hashing (with $d = 2$ in the example) is rooted at a certain node from the original

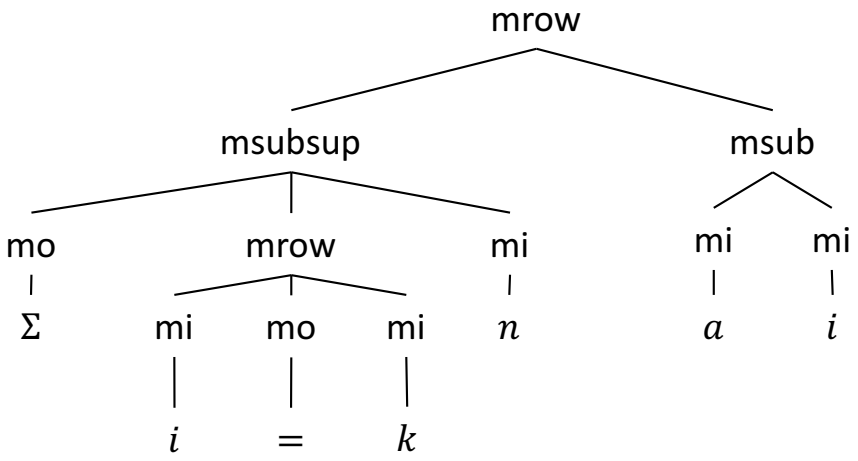


Figure 3.6: Presentation MathML of $\sum_{i=k}^{n} a_i$

tree and contains all of its descendants that are d hops away from it. Subtrees produced by SIGURE hashing is very similar to ones from subtree hashing. The main difference is that each of these subtrees has its identifiers renamed according to their appearance order. In Fig. 3.7, the third subtree produced by SIGURE hashing renames a and i into $*1*$ and $*2*$, respectively. On the other hand, the last produced subtree renames i into $*1*$, because i is the first identifier encountered by SIGURE during the traversal.

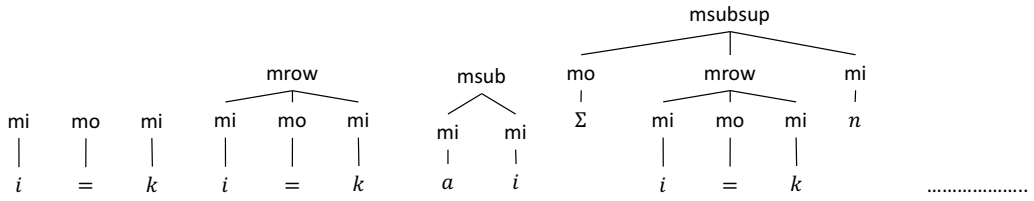
In this dissertation, we apply this hashing technique to both Presentation and Content MathML of each math expression. In this technique, each leaf of a MathML tree will be hashed based on either the literal it contains or its own tag name. The hash value of each inner node will be computed based on the hash values of its children.

3.3 Extracting Textual Information

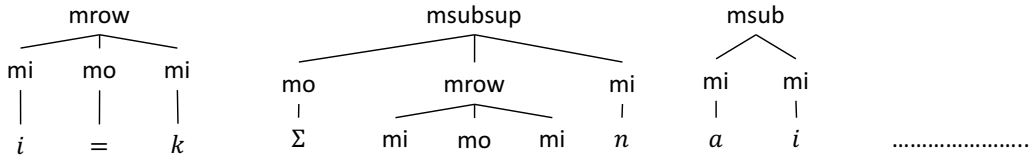
To capture the meaning of each math expression, we extract its textual information in three different levels of detail as follows.

- Math-level
  - *words in context window* are obtained by taking ten words preceding and following each math expression,
  - *descriptions* are sets of terms that precisely denote the target math expression and are automatically extracted for each math expression using a machine learning model,
  - *noun phrases* in the same sentence as the target math expression.
- Paragraph-level

Subtree Hashing:



Modular Trick Hashing (with $d = 2$):



SIGURE Hashing:

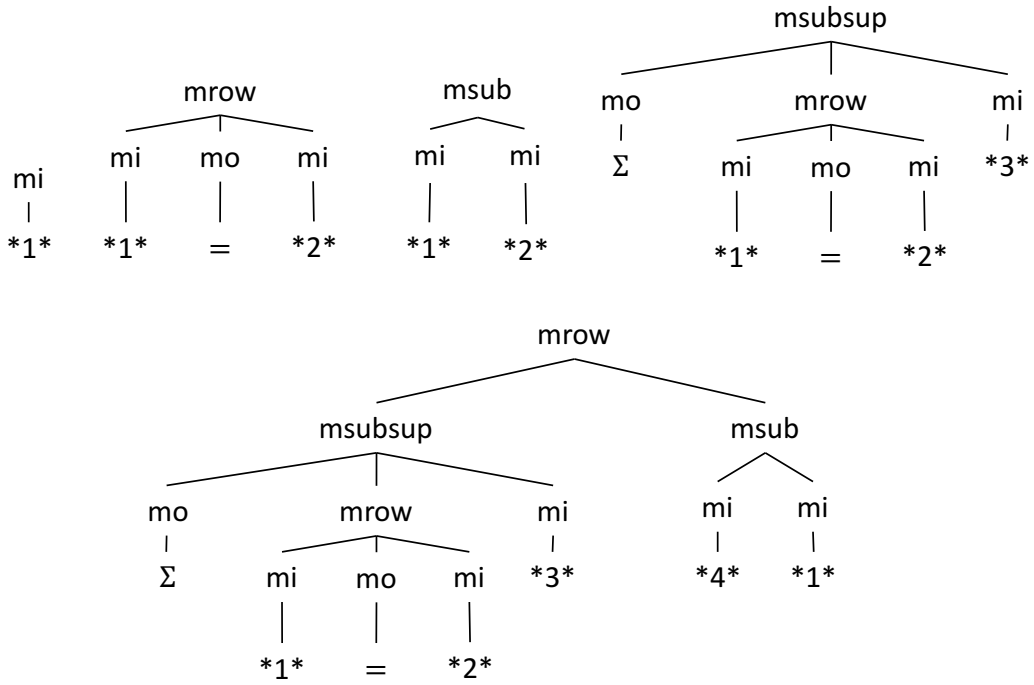


Figure 3.7: Hashing results of $\sum_{i=k}^{n} a_i$

  - *all words* in the paragraph

- Document-level
  - *title* of the document,
  - *abstract* of the document,
  - *keywords* found in the document, which are extracted using RAKE (Rose et al., 2010),
  - *descriptions* from all math expressions in the document,

– *noun phrases* in the document,

– *all words* in the document.

Except for descriptions, all these types of textual information can be easily extracted. "Words in context window" and "noun phrase" are extracted from the tokenization and parsing results of the sentence containing target math expression. "Title" and "abstract" are obtained by analyzing the tagging information in the document To extract "keywords", we use RAKE (Rose et al., 2010), and accompany each extracted keyword with a RAKE's score, which is in the interval of $[1.0, \infty)$. We later utilize the logarithm of this score, i.e. $1 + log(rakeScore)$, to boost the similarity score of each matching keyword.

Extracting Descriptions for Mathematical Expressions

The description extraction task is treated as a binary classification problem. We first pair each mathematical expression with each noun phrase that exists in the same sentence as the expression. Subsequently, for each pair, we extract several features based on sentence patterns, POS tags, parse trees, and predicate-argument structures. Next, using these features, we classify the pairs as correct (i.e. the noun phrase is the description of the expression) or incorrect.

The complete list of them are shown in Table 3.1. Features 1 to 6 are obtained by applying the six sentence patterns. $MATH$, $DESC$, and $OTHERMATH$ represent the target mathematical expression, description of the target expression, and other expressions in the same sentence as the target, respectively. Features 7 to 10 check for the existence of punctuation marks, such as colon, comma, and parentheses, and other mathematical expressions between candidate and paired expression. Intuitively, the existence of such elements in the intervening text may decrease the possibility that the candidate is a description of the expression.

Features 11 and 12 describe the location of the description candidate relative to the paired expression. Long word-distance between them may reduce the possibility that they are in the description-expression relationship. Furthermore, features 13 to 16 investigate the textual information surrounding the candidate and expression. Moreover, based on the observation over the training data, mathematical expressions and their descriptions are often separated by particular verbs usually appearing as a definitor in definition statements, e.g. denote, represent, and define. Therefore, checking the first verb that appears between the candidate and the expression is reasonable, and this is implemented as feature 17.

Table 3.1: List of machine learning features for description extraction

No.	Feature
1	... (let|set) $MATH$ (denote|denotes|be) $DESC$ ...
2	... $DESC$ (is|are) (denoted|defined|given) (as|by) $MATH$ ...
3	... $MATH$ (denotes|denote|(stand|stands) for|mean|means) $DESC$ ...
4	... $MATH$ (is|are) $DESC$ ...
5	... $DESC$ (is|are) $MATH$ ...
6	... $DESC$ ($OTHERMATH$,)* $MATH$ ...
7	If there is a colon between the desc candidate and the paired expr
8	If there is a comma between the desc candidate and the paired expr
9	If there is another math expr between the desc candidate and the paired expr
10	If the desc candidate is inside parentheses and the paired expr is outside parentheses
11	Word-distance between the desc candidate and the paired expr
12	Position of the desc candidate relative to the paired expr (after or before)
13	Surface text and POS tag of two preceding and following tokens around the desc candidate
14	Surface text and POS tag of the first and last tokens of the desc candidate
15	Surface text and POS tag of three preceding and following tokens around the paired math expr
16	Unigram, bigram, and trigram of features 15 and 13 that is combined with feature 14
17	Surface text of the first verb that appears between the desc candidate and the target math expr
18	Hop-distance (in the predicate-argument structure) between the desc candidate and the paired math expr
19	Dependencies with length of 3 hops, that is from an undirected path connecting desc candidate with paired math expr and starts from the desc candidate
20	Direction of feature 19 with respect to the candidate (incoming or outgoing)
21	Dependencies with length of 3 hops, that is from an undirected path connecting desc candidate with paired math expr and starts from the paired expr
22	Direction of feature 21 with respect to the expression (incoming or outgoing)
23	Unigram, bigram, and trigram of features 19 and 21

Prior to extraction of features 18 to 23, a graph traversal algorithm is applied to the predicate-argument structure to find the shortest path between the description candidate and the paired expression. Feature 18 then counts the graph distance between the candidate and expression. Finally, features 19 to 23 are utilizing the result from the shortest path computation to examine the dependency of a mathematical expression and paired description candidate.

To measure the performance of this description extraction method, we conduct a brief description extraction experiment on 50 mathematical papers obtained from the NTCIR Math Understanding Subtask (Aizawa et al., 2013). We set 40 papers to be the training set and 10 papers to be the test set. In addition, this experiment uses three metrics to measure the extraction performance: precision, recall, and F1-score. Furthermore, the extraction performance is measured using two different matching scenarios, namely strict matching and soft matching. An extracted description will pass the strict matching evaluation if its position, i.e. start index and length, is the same as that of a gold-standard description for the same target mathematical expression. On the other hand, an extracted description will pass the soft matching evaluation if its position contains, is contained in, or overlaps with the position of a gold-standard description for the same expression. Table 3.2 displays the results of the performance measurement.

Table 3.2: Performance of the description extraction model

Matching Scenarios	**P**	**R**	**F1**
Strict	73.72	45.88	41.40
Soft	80.80	72.77	76.58

Table 3.3 shows an example of words in context window and description. Before indexing the extracted textual information, several processes are applied to it, such as tokenizing the string, eliminating stop words and any mathematical expression, and stemming the remaining words.

Table 3.3: An example of textual information extracted for a target math expression.

Sentence: "For several applications, it is often computationally convenience to work with the natural logarithm of the likelihood function $p(x|\theta)$, called the log-likelihood."
Target math expression: $p(x|\theta)$

Type	**Example of Textual Information**
context	to work with the natural logarithm of the likelihood function called the log-likelihood
description	likelihood function

3.4 Indexing Math Expressions

We adopt Apache Solr as the search platform of our system. There are three indexes used by our system, each of which contains math-, paragraph-, and document-level information. Each index contains multiple fields, which are shown

in the Table 3.4.

Table 3.4: List of fields in each index

Index Name	**Field**
Math	p_opaths_op
	p_opaths_arg
	p_upaths_op
	p_upaths_arg
	p_sisters
	c_opaths_type
	c_opaths_arg
	c_upaths_type
	c_upaths_arg
	p_stree
	p_mtrick
	p_sigure
	c_stree
	c_mtrick
	c_sigure
	contexts
	descriptions
	noun_phrases
	contexts_depgraph (for Chapter 4)
	descriptions_depgraph (for Chapter 4)
	noun_phrases_depgraph (for Chapter 4)
Paragraph	all_words
Document	title
	abstract
	keywords
	descriptions
	noun_phrases
	all_words

The math index contains both encoded math expressions and textual information explaining the expressions. In Table 3.4, fields p_opaths_op to p_sister represent encoded math expressions using path-based encoding technique applied to MathML Presentation format, while c_opaths_type to c_upaths_arg represent the results from the same encoding technique applied to MathML Content. Fields p_stree to p_sigure and c_stree to c_sigure are the result from hash-based encoding techniques applied to MathML Presentation and Content format. The rest of fields in the math index represent the textual information found surrounding the math expressions. The textual information contained by these textual fields, together with ones in paragraph and document indexes, are described in the Chapter 3.3.

3.5 Combining All Types of Information for Searching

Most of the math search systems combine math expressions and textual keywords using Lucene scoring (Libbrecht and Melis, 2006; Munavalli and Miner, 2006; Růžička et al., 2016):

$$score(q, f) = coord(q, f) \times queryNorm(q) \times \sum_{t \in q} (tf(t, f) \times idf(t)^2 \times norm(t, f)) \tag{3.1}$$

The scores from (3.1), 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. Therefore, we need to normalize score obtained from each field. We use

$$norm(score) = \frac{score}{1 + score} \tag{3.2}$$

as the normalization function.

In addition, we suggest that different fields may have different importance. Since the proper weights will strongly depend on the implementation of math search system, we suggest that a learning method is required to obtain these weights. We consider applying learning to rank methods, which refers to machine learning techniques for constructing ranking models. There are four different learning to rank methods we take into consideration, which are briefly described as follows. Multiple linear regression estimates the weight of each feature using least square technique (Wu et al., 2011). LambdaMART (Burges, 2010) combines the strengths of boosted tree classification and LambdaRank. AdaRank (Xu and Li, 2007) linearly combines weak rankers for making ranking predictions. ListNet (Cao et al., 2007) uses different probability distributions to define the loss function.

Multiple linear regression is considered as pointwise approach, because it tries to predict the score for each single query-retrieved unit pair. LambdaMART, on the other hand, is a pairwise approach, because it transforms ranking into pairwise classification or pairwise regression. AdaRank and ListNet are listwise approaches, since they try to directly optimize the value of a chosen evaluation measures (e.g. NDCG, MAP, or P@k).

3.6 Unification

The motivation of applying unification as post-processing module in our search system is to put math expressions that can be instantiated from the query to the rank higher than the one that cannot. The unification module treats each symbol (i.e. operator, identifier, and number) found in math expressions as a constant. Only free variables found in query math expressions are treated as variables. Given two terms, each of which can be either a constant or a variable, they unify if they are the same term or if they contain variables that can be uniformly instantiated with terms in such a way that the resulting terms are equal. For instance, query $p = mv$ (all terms are constants and there is no free variable) will only unify to math expressions that have the same presentation, i.e. $p = mv$. On the other hand, query $y = 5x+?C$ ($?C$ is a free variable) will unify to $y = 5x+9$, since variable $?C$ can be instantiated to constant 9. The unification by variable instantiations is allowed when the instantiations are compatible. For an example, query $?X = ?X$ ($?X$ is a free variable) will not unify to $a = b$. This rule makes sense considering the query asks for an equality where the terms in the left- and right-hand side of the equality symbol are the same.

We implement this module using the SWI-Prolog (Wielemaker, 2015) implementation for unificatoin. To use this feature, we transform each math expression into its functor form (prefix notation). This functor form is obtained by utilizing the semantic provided by Content MathML representation of the math expression. For instance, given a math expression $a+b*c$, its functor form is: apply(plus, a, apply(times, b, c)).

We use this unification module to boost the similarity score from the math index. If a query and the math expression (or any subexpression of the expression) returned by our search system unify, we double the math-level similarity score of the expression while keep its paragraph- and document-level scores as they are.

3.7 Experiment using NTCIR-11 Math-2 Dataset

We test our math search module in NTCIR-12 MathIR task, which is the only available shared task for MIR. This task however, allows only four submissions. In addition, since we expect to achieve the best possible performance of our system, we cannot enumerate all possible settings of our system.

Prior to the NTCIR-12 MathIR task, we use the available dataset and topics from NTCIR-11 Math-2 task (Aizawa et al., 2014) to find several best settings of our system. This NTCIR-11 dataset has the same profile as the NTCIR-12 task.

They have the same corpus and the same information contained in the topics (i.e. combination of math expressions and textual keywords).

3.7.1 Objective

Here, we do a quick experiment to investigate:

- whether our score normalization works well, and
- which of the learning to rank methods (multiple linear regression, LambdaMART, AdaRank, and ListNet) performs best.

3.7.2 Dataset

The dataset released by NTCIR-11 Math-2 task consists of 105,120 scientific papers (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)

3.7.3 Experiment Settings

For this brief experiment, we use only a subset of the fields we have in our math-level database, namely five fields storing path-based encoding applied to Presentation MathML, and fields storing descriptions and context window. We suggest that this setting has already resembled our situation if we use all databases and all fields, that is we have two types of information: math encoding and textual information.

We evaluated the ranking performance of Lucene's tf-idf (specified by Eq. 3.1) and learning to rank methods for combining math expressions and textual keywords for math searching. Here, as suggested by the NTCIR-11 Math-2 requirement, we use paragraph as the retrieval unit. For Lucene scoring, the $score(q, p_i)$ of each retrieval unit (paragraph) p_i for a given query q is calculated using step 1.1. in Procedure 1. For learning to rank 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 $f_i^* = \arg\max_{f_{ij} \in p_i} score(q, f_{ij})$ 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 (3.1).

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: $s_f = norm(score(q_f, f_i^*))$ and $s_t = norm(score(q_t, f_i^*))$

   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 Precision-at-5, Precision-at-10, and Mean Average Precision (MAP), which are defined as:

$$P@n = \frac{\#(\text{relevant paragraphs retrieved in top } n)}{n} \tag{3.3}$$

$$MAP = \frac{1}{N_q} \sum_{i=1}^{N_q} MAP_i \tag{3.4}$$

$$MAP_i = \frac{1}{|R_i|} \sum_{k=1}^{|R_i|} P(R_i[k_i]) \tag{3.5}$$

where MAP_i is the mean value precision for query q_i, R_i is the set of relevant paragraphs for query q_i, $|R_i|$ refers to its size, $R_i[k_i]$ is a reference to the k-th paragraph in R_i, $P(R_i[k])$ is the precision where $R_i[k]$ paragraph is observed in the ranking of q_i, and N_q is the total number of queries. The learning to rank methods are evaluated using nested cross validation (5-inner-fold for tuning hyperparameters and 10-outer-fold for reporting performance).

Table 3.5: Performance of reranking method in NTCIR-11 Math-2 task

$^{1-9}$ 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	.4108	.4440	.3820	.5085	.6800	.6220
2.	keywords	.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.	Linear Regr.	**.6259**^{123}	.6520^{12}	**.5660**^{13}	.7471^{123}	.9520^{13}	.9140^{13}
5.	λMART	.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}
$s_t + s_f$		.6245^{123}	.6560^{12}	.5660^{13}	.7248^{123}	.9360^{13}	.8720^{13}

3.7.4 Experiment Result

Table 3.5 shows the ranking performance from learning to rank, Lucene scoring (without score normalization), and searching using either only math or keywords as queries. First, the learning to rank methods outperform Lucene scoring approach at all metrics. The highest performance obtained by learning to rank 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 learning to rank methods give improvements over text-only search in both relevancy settings. Among the learning to rank 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. In addition, unlike the other learning to rank methods, the linear regression method does not need to be implemented as a post-process reranking module. We can modify the queries we send to Solr to reflect the weights we want to set in the linear regression. Thus, we have a simpler pipeline for our search system.

The final row of Table 3.5 shows the performance of applying score normalization without any weighting. This approach surprisingly performs well. The produced precision is close to the performances of the learning to rank methods.

Conclusion

From the results given above, we decide to implement the score normalization in the next experiment. Furthermore, because of all the benefits of linear regression, we prefer multiple linear regression to the other learning to rank methods for our NTCIR-12 MathIR submissions.

3.8 Evaluation in NTCIR-12 MathIR

3.8.1 Objective

The objective of this experiment is to investigate the effectiveness of multiple linear regression and unification modules in our math search system.

3.8.2 Dataset

The corpus for this experiment is the same as the one from previous experiment, namely 8,301,578 retrieval units (paragraphs) that contain around 60M math expression. There are 29 topics in this task.

Formulae Query Variables (Wildcards). Formulae in the query may contains query variables (`<qvar>`) that act as wildcards, which can be matched to arbitrary subexpressions on the retrieved formulae. Query variables are named and indicated by a question mark (e.g. $?v$). To handle this, we make several adjustments to our math search pipeline as follows.

- *Path-based encoding module.* This module handles each wildcard found in the queries by simply omitting any vertical paths, i.e. `opaths` and `upaths` directing to it and excluding it from any `p_sisters`.
- *Hash-based encoding module.* The algorithms in this module generate hash values based on the name of the wildcard. For instance, given the query variable $?v$, this module produce a hash value from literal v.
- *Unification.* The wildcards are the only elements of the queries that are treated as free variables.

3.8.3 Experiment Setup

There are four main features we apply for obtaining the ranked list, which are as follows.

- score normalization

- use cold-start weights obtained by multiple linear regression
- apply unification to boost the math-level score
- utilize textual information obtained from exploiting dependency graph (*this will be explained in Chapter 4*)

The list of features we used for each submission in this task is depicted by Table 3.6. Even though dependency graph is the subject of the Chapter 4, we include it here because the evaluation of the linear regression and unification modules involves our submissions that utilize dependency graph. We will however, limit the involvement of dependency graph in our discussion here.

Table 3.6: Features used in each NTCIR12-MathIR submission

Run ID	**norm.**	**dep. graph**	**cold-start**	**unif.**
nodep_lr_unif (nd-lr-u)	O		O	O
allfields_nowgt_unif (a-nw-u)	O	O		O
allfields_lr (a-lr)	O	O	O	
allfields_lr_unif (a-lr-u)	O	O	O	O

To obtain the set of cold-start weights, we construct a training set using the relevance judgment result released by the NTCIR-11 Math-2 task (Aizawa et al., 2014). The procedure for the training set construction is very similar to the Procedure 1 in the previous experiment. The only difference is that now we set weights for all fields we have (see Table 3.4), instead of only two fields. The Procedure 2 specifies the complete steps to construct the training set. This differs from Procedure 1 at the Step 2.

Procedure 2 (Training Set Construction for NTCIR-12 MathIR)

1. Get a math expression f_i^* to represent each retrieval unit (paragraph) p_i in the training set.
   1.1. Set $f_i^* = \arg\max_{f_{ij} \in p_i} score(q, f_{ij})$ 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 Lucene's default TF-IDF similarity with the subscore from each field being normalized using Eq. 3.2.
2. Get features for each retrieval unit p_i in the training set.
   2.1. For each topic, compose a query q_{field} for each field we have in all indexes.

2.2. Features: $s_{field} =$

$$\begin{cases} score(q_{field}, f_i^*) \text{ for } field \text{ in math index} \\ score(q_{field}, p_i) \text{ for } field \text{ in paragraph index} \\ score(q_{field}, d_i) \text{ for } field \text{ in document index} \end{cases}$$

where d_i is the document where p_i lies.

2.3. Response: relevancy score (0 to 4) provided in judgment result.

For the searching procedure, both score normalization and cold-start weights are specified in the query using FunctionQuery provided by Solr. We obtain initial ranked lists of 2,000 units from math, paragraph, and document indexes. For the math index, we expand the ranked list in increments of 2,000 until the number of unique paragraphs referred to by the list is at least 2,000.

3.8.4 Experiment Results and Discussion

The performance of our system in this task, which is measured using trec_eval, is shown in Table 3.7. Our system achieves the best precision among other teams.

Table 3.7: Search performances in NTCIR-12 MathIR

RunID	**P@5**	**P@10**	**P@15**	**P@20**
Relevant				
nodep_lr_unif (nd-lr-u)	.2345	.1966	.1747	.1586
allfields_nowgt_unif (a-nw-u)	.2828	.2379	.2184	.1948
allfields_lr (a-lr)	.2552	.2379	.2092	.1828
allfields_lr_unif (a-lr-u)	.2621	.2448	.2046	.1810
ICST (Gao et al., 2016)	.2276	.1862	.1632	.1362
MIRMU (Růžička et al., 2016)	.1241	.1345	.1218	.1069
RITUW (Davila et al., 2016)	.2069	.1517	.1126	.0948
SMSG5 (Thanda et al., 2016)	.0690	.0931	.0874	.0810
Partially Relevant				
nodep_lr_unif (nd-lr-u)	.4828	.4793	.4690	.4552
allfields_nowgt_unif (a-nw-u)	.5448	.5345	.5149	.4897
allfields_lr (a-lr)	.5586	.5379	.5034	.4690
allfields_lr_unif (a-lr-u)	.5586	.5483	.5126	.4707
ICST	.5517	.4966	.4299	.4000
MIRMU	.3931	.3655	.3402	.3207
RITUW	.4966	.3966	.3310	.2879
SMSG5	.3517	.3724	.3586	.3397

Here, we will not use our allfields_* submissions to compare our system to other participating systems. We defer the explanation for the Chapter 4. We will use them only for comparing the effectiveness of cold-start weight (all_fields_nowgt _unif v. all_fields_lr_unif) and unification module (all_fields_lr_unif v. all_fields_lr).

The performance of our system that utilize fields shown in Table 3.4 is given by the submission nodep_lr_unif (nd-lr-u). We examine the effectiveness of the components as follows.

Cold-Start Weight (Learning to Rank by Multiple Linear Regression) does not perform well enough. The submission a-lr-u, which uses cold-start weights, does not outperform a-nw-u consistently. It outperforms a-nw-u only in three occasions, namely P@5 (in partial relevance judgment) and P@10 (both in relevance and partial relevance), while the opposite happens in the other five occasions. The underperforming cold-start weights indicates that the linear regression model learned from NTCIR-11 Math-2 dataset does not generalize well. This may happen due to several reasons. First, the relevance assessors, which are third-year and graduate students of mathematics, in Math-2 and MathIR tasks may have different criteria in determining if a retrieval unit is relevant to a given query. While an assessor may expect a relevant paragraph to simply contain math expression with presentation similar to the query, another assessor may be less strict with the presentation but expect the context of the retrieved paragraph matching the query keywords. Moreover, in these two tasks, no specific instructions were given as to how the relevance was to be judged. Thus, the assessors had to rely on their mathematical intuition, the described information need, and the query itself.

The second reason is the multicollinearity issue in the regression model. The introductions of too many fields that are correlated to each other causes this issue. This did not appear in our previous experiment (see Chapter 3.7, because we utilized coarser granularity of scores, i.e., math and text similarities. Multicollinearity is less likely to happen when there are less variables. Therefore, a possible solution is to reduce the number of the variables (i.e. field scores) by grouping them into more meaningful entities, such as, math similarity score, text similarity in math-level database, paragraph-level similarity, and document-level similarity. This may help us to eliminate the multicollinearity issue and also to better understand the importance of each field.

We can also reconsider the implementation of cold-start weight by examining other learning-to-rank algorithms, such as SVM^{rank} (Joachims, 2006) and ListNet (Cao et al., 2007), in handling our various fields. These algorithms however, will add complexity to the math search pipeline, since we will have to implement them as a post-processing module.

Table 3.8: System configurations in NTCIR-12 MathIR

Team	Pres. MathML	Cont. MathML	Query Variables (Unif.)
Our System	YES	YES	YES/NO
ICST	YES	NO	YES
MIRMU	YES	YES	NO
RITUW	YES	NO	YES
SMSG5	YES	NO	YES

Unification performs well enough. The submission a-lr-u outperforms a-lr in most of the measurements. At P@15 and P@20, unification performs well for partial relevance, but does not for relevance judgment. This suggests that unification module obtains several partially relevant math expressions that were previously outside of the top-20 and boosts their scores. As a consequence, these expressions may now be ranked higher than several highly-relevant expressions that were previously already in the top-20. If these highly-relevant expressions (and all of its subexpressions) and the query do not unify, then they may be now out of the top-20.

Feature Comparison with Other Systems. For further analysis, we observe the participant system configuration as shown in Table 3.8, which is compiled by the task organizer (Zanibbi et al., 2016). We suggest that the main reason why our system can significantly outperforms most other systems are the use of multiple types of textual information and score normalization. We do not think our use of unification to overcome query variables is the main reason because other systems, such as ICST , RITUW, and SMSG5, also handle the query variables.

- *Multiple types of textual information.* Unlike ours, other systems do not extensively utilize textual information. They simply store one or two types of textual information, e.g., all words in each paragraph or text surrounding math expressions. As a result, we can outperform all systems at all measurements in highly relevant judgment. We can have better P@5 than others, because of the use of the math-level textual information, which is designed to help improve precision of our system. If we limit our analysis up to the top 20 retrieval units, the result of P@20 indicates that our use of paragraph and document level text enable our system to have higher recall than other participants. We found that this performance of our system is impressive, considering that the second place system, ICST, perform extensive reranking process using 255 dimension features.

In the partially relevant judgment, ICST perform better than us at P@5 and P@10 measurements. We suggest that their reranking procedure is the key reason here. However, next, our system is better at P@15 and P@20. This again shows that our paragraph and document level textual information allows our system to have a good recall, that is by retrieving more relevant retrieval units. Given this analysis, we suggest that our multiple types of textual information is one of the key features of our system.

- *Score Normalization* in Eq. 3.2 that we applied during searching allows our system to utilize many fields for searching without any concern regarding whether scores from certain fields will improperly dominate the final score, especially when the number of terms generated for each of these certain fields is much larger than the number of terms generated for other fields. As a result, the submission a-nw-u of our system (no weighting, but with normalization) can finish at the top in this task.

We also observe that the use of Content MathML may not be useful enough. ICST and RITUW, which do not use Content MathML, give performances close to ours. We suggest that this happens because the tool (Miller, 2016) used in this task to produce content markup does not always return correct Content MathML. When this tool cannot read the semantic of a math expression (hence the wrong content markup), it will produce a content markup that expresses the math expression in the way similar to presentation markup, i.e., how the expression is visually rendered. Furthermore, this tool can give correct content markup for simple math expressions, but not for the complex ones or when there is a mistake in the math expression. Yet, the corpus in this task contains scientific papers from arXiv, which majority of them include long and complex math expressions.

Some examples of imperfect Content MathML are shown in Fig. 3.8. Both of the obtained Content MathML contains error elements `<cerror>`, denoting one or multiple mistakes detected in the math expressions. As a consequence, these content markups express the math expressions in infix notation, similar to the presentation markup.

3.9 Space Utilization and Processing Time

The execution environment of our system is two Linux servers, each of which has 64 processors. We use one server to store math indexes, and another server to store paragraph and document indexes. The sizes of math, paragraph, and document indexes are 426.39GB, 898.18MB, and 1.26GB, respectively.

$$F = \begin{cases} 1 & \text{iff word is in sentence} \\ 0 & \text{else} \end{cases}$$

Expected Content MathML	Obtained Content MathML

`<math>`
`  <apply>`
`  <eq/>`
`  <ci>F</ci>`
`  <piecewise>`
`  <piece>`
`  <cn>1</cn>`
`  </piece>`
`  <piece>`
`  <cn>0</cn>`
`  </piece>`
`  </piecewise>`
`</apply>`
`</math>`
	`<math>`
	`  <cerror>`
	`  <csymb>fragments</csymb>`
	`  <ci>F</ci>`
	`  <eq/>`
	`  <cerror>`
	`  <csymb>fragments</csymb>`
	`  <ci>normal-{</ci>`
	`  <mtext>`
	`  1iffwordisinsentence0else`
	`  </mtext>`
	`  </cerror>`
	`</math>`

$a > b < c$) (a wrong extra parentheses in the end)

Expected Content MathML (of $a > b < c$)	Obtained Content MathML

`<math>`
`  <apply>`
`  <and/>`
`  <apply>`
`  <gt/>`
`  <ci>a</ci>`
`  <ci>b</ci>`
`  </apply>`
`  <apply>`
`  <lt/>`
`  <share/>`
`  <ci>c</ci>`
`  </apply>`
`  </apply>`
`</math>`
	`<math>`
	`  <cerror>`
	`  <csymbol>fragments</csymbol>`
	`  <csymbol>a</csymbol>`
	`  <gt/>`
	`  <csymbol>b</csymbol>`
	`  <lt/>`
	`  <csymbol>c</csymbol>`
	`  <ci>normal-)</ci>`
	`  </cerror>`
	`</math>`

Figure 3.8: Examples of wrong content markup. Both cases express the presentations, not the semantics, of math expressions.

In addition, the processing time required by our system for each submission is shown in Figure 3.9. The first three plots depict the time needed to obtain initial ranked lists from document, paragraph, and math indexes, respectively. For each plot, the first four boxplots correspond to our four submissions. These submissions utilize score normalization step and some of them use also cold-start weights. As a comparison, we put the fourth boxplot "all (default)," which denotes the query time when none of these two features are implemented.

The order of the query time from the shortest to the longest is document, paragraph, and then math index. Two factors influencing the query time are:

- the more retrieval units an index stores, the longer the query time is, and
- the more query terms and fields we specify for searching in a certain index,

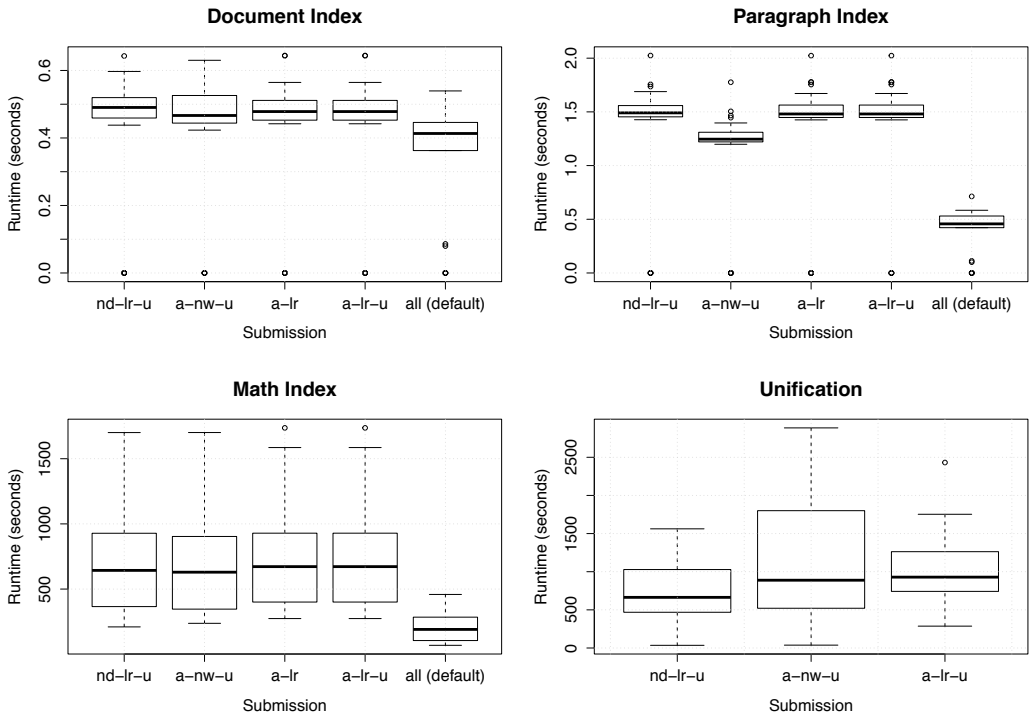


Figure 3.9: Query time of our math search system in the NTCIR-12 MathIR task

the longer the query time is.

Among our four submissions, the a-nw-u has the shortest query time. This happens because without the use of cold-start weights, the query (i.e. FunctionQuery) becomes simpler and less calculation is performed by Solr. This reasoning also explains why all our four submissions require query time that is significantly longer than the "all (default)" case. All our four submissions use FunctionQuery to apply score normalization to each field specified in the query. The more fields a query specifies, the more score normalization step Solr has to perform, thus the longer the query time is. This observation shows that the positive impact of score normalization procedure, i.e. ensures that none of the scores from certain fields will improperly dominate the final score, comes at the cost of the query time.

The last plot in Fig. 3.9 displays the wall-clock time taken by our system to complete the unification procedure that is parallelized into 50 processes. All four submissions have the same median of unification time. However, the time distribution of submission a-nw-u is more skewed than the other two submissions. Since all these submissions contain the same number of math expressions in the initial ranked list, this difference in skewness indicates that the initial ranked list of a-nw-u contains more complex math expressions, i.e. have deeper levels of MathML representations and contain more sub-expressions, than the ranked lists from nd-lr-u and a-lr-u. As unification step is performed between math query and

each sub-expression of the retrieved math expressions, the more sub-expressions a math expression contains, the longer the unification time is.

3.10 Conclusion

We have presented the detail of our math search system. We implemented two types of encoding technique (path- and hash-based), several types of textual information to cover three levels of information granularity (math expression, paragraph, and document levels), score normalization, cold-start weights, and unification.

Among our submissions, we found out that score normalization and unification are integral parts of our search system. The cold start weights, however, do not have a good impact on the search performance due to a possible multicollinearity issue. Furthermore, by comparing to the other available math search systems, we suggest that our multiple types of textual information are useful and the key features in our system.

A possible future work is to set the unification module to vary the boosting score based on the depth location of the subexpression inside the retrieved math expression to which the query unifies. By doing so, however will also increase the search time, since unification is an expensive feature. The effectiveness of cold-start weights should also be reconsidered if we remove multicollinearity issue by grouping the field scores into more meaningful entities. In addition, an application of other learning to rank algorithms can also be further examined.

Chapter 4

Utilizing Dependency Relationships between Mathematical Expressions in Mathematical Search System

(This chapter is censored because it is under review in Information Retrieval Journal)

Chapter 5

Linking Mathematical Expressions to Wikipedia

5.1 Introduction

Entity linking (EL) is the task of linking entity mentions in text to corresponding entities in a knowledge base. One variant of entity linking is wikification (Bunescu and Paşca, 2006; Cucerzan, 2007; Mihalcea and Csomai, 2007; Milne and Witten, 2008; Ferragina and Scaiella, 2010; Han et al., 2011; Ratinov et al., 2011; Cheng and Roth, 2013; Guo and Barbosa, 2014), which identifies a set of entity mentions in a document and then locates the most accurate mapping from these mentions to corresponding Wikipedia articles. Current work on wikification has focused on natural language mentions, often expressed in the form of noun phrases. Such an approach, however, may not be sufficient for the wikification of technical documents.

Important concepts in technical documents are often found not only in the form of natural language text, but also in mathematical expressions. Therefore, to *wikify* all important concepts in scientific documents, especially those from science, technology, engineering, and mathematics (STEM), we need to consider both forms. To the best of our knowledges, there is no previous study has attempted to wikify math expressions found in technical documents. Hereinafter, we refer to the task of wikifying math expressions as math entity linking (MEL).

The possible applications of MEL include computer-assisted learning, unsupervised concept graph generation (Agrawal et al., 2012), and document representation (Ni et al., 2016). For example, in computer-assisted learning, MEL enables us to develop an educational application that can suggest supplementary information for math formulae unknown to a reader. In unsupervised concept graph generation, one significant challenge is to identify informative concepts within documents. We suggest that the important concepts in a scientific document are

often denoted with math expressions; thus, we can exploit math expressions to identify these concepts.

The challenges we encountered in developing an MEL system are as follows.

1. The semantics of math expressions are often difficult to identify from their surface level representations because of their abstract notation (Kohlhase and Sucan, 2006). This challenge is shared by MEL task and math formula search.

2. Math expressions that match a given math mention are likely to be important in the containing Wikipedia articles. This is a unique challenge that appears in MEL, but not in math searches, which arises because the nature of the wikification task is to assign Wikipedia titles to mentions. Hence, we expect that given a math mention, an MEL system will return a Wikipedia article that contains math expression(s) similar to the mention and, more importantly, whose title precisely describes the math mention.

3. There is no available MEL dataset that is large enough to allow us to implement a supervised-learning-based MEL system.

5.2 Problem Definition

We formalize the problem of linking mathematical expressions to Wikipedia as follows. Given a document d containing a set of math expressions $M = \{m_1, \ldots, m_n\}$, our goal is to assign each math mention m_i a Wikipedia article t_i. This problem formulation is analogous to that of a wikification task. The set of important math expression M is comparable to the set of mentions in wikification.

The overview of our MEL framework is given in Algorithm 1. In this study, we make an assumption as follows.

Assumption 1 *The set of math mentions M has already been discovered.*

Since we further assume in this paper that all math expressions (math mentions and math expressions in the knowledge base) are in MathML format, we can detect math mentions from a document by simply identifying the appearances of `<math>` tag in the document. We take mention detection to be outside this study's scope and instead focus on (i) enriching math mentions, (ii) generating candidates for each mention, and (iii) disambiguating these candidates.

Figure 5.1 shows an example of our framework and demonstrates the requirements of an MEL task. The matching math expressions for a given mention must be important in the containing documents. The Wikipedia article on Maxwell's_equation is the correct link for the math mention $\nabla \times \mathbf{B} = \mu_0\mathbf{J} + \mu_0\epsilon_0\frac{\partial}{\partial t}\mathbf{E}$. The other articles, such as Weibel_instability and Magnetic_field, are not the correct answers. Although these articles contain math expressions that are similar to the mention, they are not the representatives of the articles in which they appear. In the Weibel_instability article, the matching expression appears only to derive a simple example of Weibel instability. In the Magnetic_field article, the matching math expression appears to explain relationships between magnetic (the main concept in the article) and electric fields.

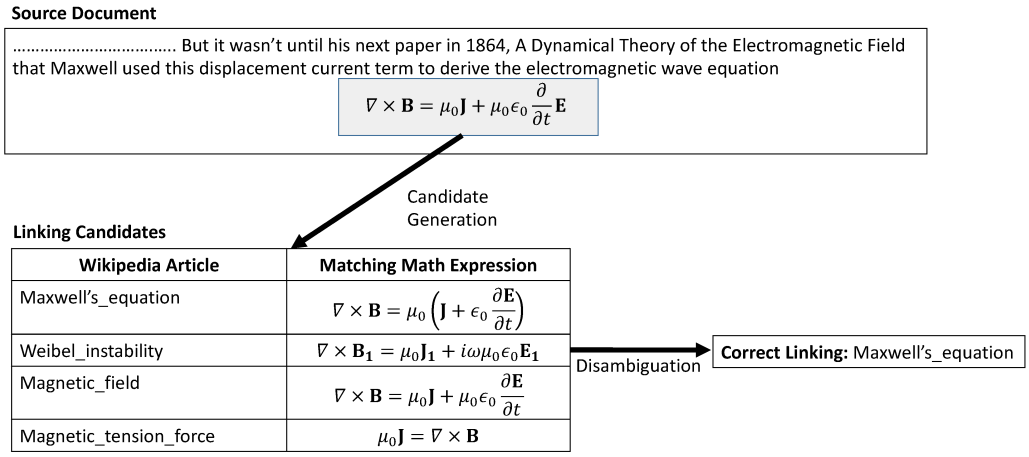


Figure 5.1: Producing a correct link for equation $\nabla \times \mathbf{B} = \mu_0\mathbf{J} + \mu_0\epsilon_0\frac{\partial}{\partial t}\mathbf{E}$

Algorithm 1: Math Entity Linking Framework

Input: document d with mentions $M = \{m_1, \ldots, m_n\}$
Output: links $\Gamma = \{t_1, \ldots, t_n\}$

1 **Mention Enrichment:** For each $m_i \in M$, construct a tuple $(\mathbf{m_i}, text_i)$ where $\mathbf{m_i}$ is the subexpressions extracted from m_i and $text_i$ is the textual information of m_i
2 **Candidate Generation:** For each mention m_i, generate link candidates $T_i = \{t_{i,1}, \ldots, t_{i,k}\}$ by exploiting the enrichment results
3 **Disambiguation:** Find a link solution $\Gamma = \{t_1, \ldots, t_n\}$ where $t_i \in T_i$ is the best non-null link for m_i

5.3 Mention Enrichment

In the enrichment module, our system takes the input set of math mentions M and performs math and text enrichment to each math mention m_i. The output

of the enrichment module for a given math mention m_i is $(\mathbf{m_i}, text_i)$, where $\mathbf{m_i}$ and $text_i$ are the results of math and text enrichment, respectively.

5.3.1 Math Enrichment

A given math mention m_i may describe (in)equalities. This enrichment module finds the top-level (in)equality relation symbols, and then splits the math mention on these symbols. The output of this step is a set $\mathbf{m_i} = \{m_i, sub_i^1, sub_i^2, \dots\}$, which contains the original math mention m_i and a set of split subexpressions $\{sub_i^1, sub_i^2, \dots\}$.

For an example, given math mention m_i

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}},$$

the returned set $\mathbf{m_i}$ contains m_i and the following three subexpressions:

- r_{xy}
- $\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)s_x s_y}$
- $\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2 \sum_{i=1}^{n}(y_i-\bar{y})^2}}$

This math enrichment technique is also applied to the math expressions we store in our database.

5.3.2 Text Enrichment

For each given math mention m_i, this module returns $text_i = concat(desc_i, np_i)$, which is a bag-of-words representation of the textual information in m_i. It is obtained by concatenating the descriptions $desc_i$ and the concatenated noun phrases np_i of m_i. Each description, which is a phrase in document d that refers directly to m_i, is automatically extracted using a machine learning based model (see Chapter 3.3). On the other hand, noun phrases are extracted by parsing the sentences that contain m_i. We further enrich these two types of textual information using dependency graph (Chapter 4), that is we also include the noun phrases and descriptions of each symbol found within m_i in the $desc_i$ and np_i, respectively.

5.4 Candidate Generation

To identify Wikipedia articles t_i that may correspond to math mention m_i, the enrichment results $\mathbf{m_i}$ and $text_i$ are used to construct query q_i. The candidate generation module outputs a ranked list of Wikipedia articles that contain math expressions similar to the math mention in both presentation and meaning. In addition, as a requirement of the MEL task, we must ensure that the math mention can be described by the titles of the Wikipedia articles. Due to this constraint, we measure not only the math-level similarity, but also the document-level similarity for the link candidates.

5.4.1 Math-Level Similarity

The similarity between math mention m_i and a math expression v in a knowledge base is calculated as follows:

$$sim_m(m_i, v) = sim_{math}(m_i, v) + sim_{text}(m_i, v) \tag{5.1}$$

To measure the presentation similarity between m_i and v, we first encode these math expressions using path-based and hash-based math representations that were described in Chapter 3.2. This encoding process produces several features for each input math expression. The function sim_{math} exploits the features generated for m_i and v, and then measures the tf-idf similarity ϕ_{mf} for each feature mf.

$$sim_{math}(m_i, v) = \max_{m' \in \mathbf{m_i}} \sum_{mf} \phi_{mf}(m', v) \tag{5.2}$$

To measure the semantic similarity between math expressions, we use sim_{text} to measure the tf-idf similarity of their textual information. Function sim_{text} utilizes the textual information $text_i$ of math mention m_i and $text_v$ of math expression v. Textual information $text_i$ and $text_v$ contain explanations of not only their corresponding math expressions, but also several symbols found within these expressions. Therefore, sim_{text} measures not only the semantic similarity between m_i and v, but also the agreement between textual information about symbols in m_i and symbols in v.

5.4.2 Document-Level Similarity

The document-level similarity between math mention m_i and a math expression v found in Wikipedia article t is calculated as follows.

$$sim_d(m_i, v) = \sum_{df} \phi_{df}(text_i, t) \tag{5.3}$$

Function sim_d measures the tf-idf similarity ϕ_{df} between textual information $text_i$ from a math mention and each textual feature df of the Wikipedia article $t_{i,j}$. The textual features of each Wikipedia article include its title, summary, body, anchor text, weighted keywords (Rose et al., 2010), and text (i.e., descriptions and noun phrases) surrounding the math expressions found in the article.

5.5 Disambiguation

For disambiguation purposes, our system reranks the link candidate list T_i obtained in the previous step. Prior to reranking, we extract two disambiguation features for each returned candidate, namely the match location and the importance of the math expression. Disambiguation utilizes these two features in addition to the math-level and document-level similarity measures obtained in the previous step.

5.5.1 Matching Location

Given two math expressions v_1 and v_2, we define a feature $loc(v_1, v_2)$ that determines if math expression v_1 or any of its split subexpressions can be found in v_2; if so, this feature returns a score that reflects the corresponding location, weighted by the similarity between v_1 and v_2.

This feature is described in Algorithm 2. We first apply math enrichment to v_1 and v_2 (lines 3–4) to obtain $\mathbf{v_1}$ and $\mathbf{v_2}$. Given these enrichment results, we determine if any split subexpression of $v_1' \in \mathbf{v_1}$ can be found in v_2 (line 5). If there are several split subexpressions of v_1 that appear in some split subexpressions of v_2, we obtain the pair of split subexpressions v_1' and v_2' that maximize the scoring function $match(v_1', v_2')$.

The function $match(v_1', v_2')$ counts the weighted frequency of v_1' appearing in v_2'. This is reflected by the use of indicator function $I(\cdot)$ on line 8. Each appearance of v_1' in v_2' is weighted by the location at which v_1' appears in v_2'. The deeper the subtree location of v_1' in v_2', the lower the assigned weight ($\frac{1}{1+depth_{2,j}}$). In addition, the weight $\frac{|nodes(v_1')|}{|nodes(v_2')|}$ reflects the ratio of the number of XML elements

found in the MathML Presentation format of v_1' to that of v_2'. The output of function $match(\cdot, \cdot)$ is a number in the range [0..1]; an output of 1 expresses that v_1' perfectly matches v_2'.

Algorithm 2: Defining $loc(v_1, v_2)$ feature

Function $loc(v_1, v_2)$
  $//MathEnrich()$ is defined in Chapter 5.3
  $\mathbf{v_1} = MathEnrich(v_1)$
  $\mathbf{v_2} = MathEnrich(v_2)$
  return $\max_{\substack{v_1' \in \mathbf{v_1} \\ v_2' \in \mathbf{v_2}}} match(v_1', v_2')$
Function $match(v_1',\ v_2')$
  $\{(depth_{2,1}, v_{2,1}'), \ldots, (depth_{2,r}, v_{2,r}')\} = ExtractSubtrees(v_2')$
  return $\frac{|nodes(v_1')|}{|nodes(v_2')|} \sum_j I(v_1' = v_{2,j}') \frac{1}{1+depth_{2,j}}$

5.5.2 Importance of Math Expressions

The second disambiguation feature is the importance measure of math expression v contained in link candidate t. This feature, together with sim_d, is used to ensure that the title of link candidate t can describe the meaning of the math mention. Algorithm 3 describes the procedure for obtaining this feature.

Algorithm 3: Defining the Importance of Math Expressions

Input: a set of math expressions $V = \{v_i\}$ in a Wikipedia article
Output: a vector of importance scores $\boldsymbol{w}$

Generating a Math Expression Dependency Graph: Obtain a directed graph $G(V, E)$
Applying Personalized PageRank to $G(V, E)$**:** Obtain a vector $\boldsymbol{c}$, where c_i represents the PageRank estimate of v_i in article
Clustering: Cluster the vertices in the dependency graph. All math expressions in cluster C will share the same weight: $weight(C) = \sum_{v_i \in C} c_i$
return importance vector $\boldsymbol{w}$, where the weight w_i of each math expression v_i is equal to $weight(C)$ of cluster C into which v_i is clustered.

Generating Dependency Graph.

We generate dependency graph using our heuristic method that was described in the Chapter **??**.

Applying Personalized PageRank to a Dependency Graph

We obtain the importance measure of each math expression (i.e., each vertex in the dependency graph) by applying the Personalized PageRank algo-

rithm (Haveliwala, 2003) to the dependency graph. Prior to applying the PageRank algorithm, we flip the directions of all edges in the dependency graph. Hence, a directed edge e_{12} from vertex (math expression) v_1 to v_2 now indicates that the string representation of the Presentation MathML-formatted v_1 contains the string representation of the Presentation MathML-formatted v_1. The idea here is that a math expression is considered to be important in a document if there are many math expressions in the document appearing as its symbols or its subexpressions.

The Personalized PageRank algorithm exploits the linkage structure of the graph to compute the score of each vertex. The higher the score of a vertex, the more important it is. Let A be the transition matrix of the dependency graph, with A_{ij} being the probability of reaching vertex v_j from v_i, which can be computed as follows:

$$A_{ij} = \frac{1}{out\_degree(v_i)} \tag{5.4}$$

The iterative step, in which we compute a new PageRank vector estimate $\boldsymbol{c'}$ from the current PageRank estimate $\boldsymbol{c}$ and the transition matrix A is

$$\boldsymbol{c'} = \beta A\boldsymbol{c} + (1-\beta)p \tag{5.5}$$

where p is a preference vector used to avoid sinks and guarantee convergence. We define the preference vector p as follows:

$$p_i = \frac{pref_i}{\sum_i pref_i} \tag{5.6}$$

There are two factors we consider when establishing preference scoring function $pref_i$, namely the location and display of math expression v_i in the article. Based on our observation, important math expressions often appear in the early sections of a Wikipedia article, since later sections usually contain derivations or applications of the primary math expressions. Furthermore, important math expressions are frequently displayed on their own lines. To reflect this, we set the scoring function $pref_i$ as follows.

$$pref_i = \begin{cases} \frac{1}{1+section(v_i)} & \text{if } display(v_i) \neq inline \\ 10^{-5} & otherwise \end{cases} \tag{5.7}$$

in which function $section(v_i)$ returns the section number of the location of v_i in the article. If v_i appears in the summary prior to the table of contents, $section(v_i)$ returns 0.

PageRank over Weighted Dependency Graph. PageRank was originally designed to exploit only graph structures. Later, several studies, such as TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004), applied PageRank to weighted graphs. For instance, LexRank introduced a similarity graph between sentences. Each vertex represents a sentence, and each edge has a weight representing the similarity between two connected sentences. These studies motivate us to apply the PageRank algorithm to weighted dependency graphs. Here, we set $loc(v_1, v_2)$ as the weight of edge e_{12} (from math expression v_1 to v_2).

Clustering

We then cluster vertices V in dependency graph $G(V, E)$. Each constructed cluster C is defined as

$$C = \{v_i | (\forall a, b)[v_a, v_b \in C \wedge e_{ab}, e_{ba} \in E \wedge a \neq b]\}.$$

Each cluster C is assigned a weight that is equal to the sum of the PageRank estimates of the contained math expression: $weight(C) = \sum_{v_i \in C} c_i$. Finally, instead of taking c_i, we take the $weight(C)$ of cluster C, into which v_i is clustered, as the importance score w_i of v_i. We adopt this strategy because a distinct math expression may appear multiple times in an article. By using this approach, given two math expressions with the same PageRank estimates, the one that appears more often in the source article will have a higher importance weight.

As an example of the importance measurement, we obtain the PageRank (i.e., the importance score) of each math expression contained in the Wikipedia article titled "Supervised learning" using the unweighted dependency graph. An excerpt of the result is shown in Table 5.1.

Table 5.1: A list of math expressions from the Wikipedia article titled "Supervised_learning" in descending order of importance score.

Math Expression	Meaning	Impt. Score
$J(g) = R_{emp}(g) + \lambda C(g)$	cost function	0.4324
$L(y_i, \hat{y})$	loss of predicting the value $\hat{y}$	0.3360
$P(y\|x)$	probabilistic model of learning algorithms	0.1066
$R(g)$	risk of function g	0.0682
...	...	...

Learning to Rank

The final step in the disambiguation process is to rerank the list T_i of link candidates using the features extracted for each candidate $t_{i,j} \in T_i$. The list of features we use in this study are displayed in Table 5.2. Our system utilizes LambdaMART (Wu et al., 2010; Burges, 2010) as its reranker. LambdaMART is trained using data collected from the NTCIR-12 MathIR Wikipedia task (Zanibbi et al., 2016). The method for compiling the training data is described in the next section.

After obtaining the reranking model, we can apply it to list T_i and acquire the top link, $t_i \in T_i$, which is the best non-null link for mention m_i.

Table 5.2: Features for Learning to Rank.

math-level similarity	:	$sim_m(m_i, v)$
doc-level similarity	:	$sim_d(m_i, v)$
matching location	:	$loc(m_i, v)$ and $loc(v, m_i)$
math importance	:	w_v^1 (importance score using unweighted dependency graph) and w_v^2 (importance score using weighted dependency graph)

5.6 Experiment

5.6.1 Dataset

We evaluate our MEL framework on a dataset constructed from the NTCIR-12 MathIR Wikipedia subtask (Zanibbi et al., 2016). Unlike the dataset we used in the previous chapters, which provides scientific documents as corpus, this subtask used the Wikipedia articles as corpus. The Wikipedia articles were initially in MediaWiki format. The NTCIR organizers extracted the math expressions by first converting the MediaWiki math templates to LaTeX and then converted them together with LaTeX formulae demarcated by `<math>` tags in the articles to MathML format using LaTeXML (Miller, 2016).

We index all math expressions and the textual information found in the Wikipedia dataset in a configuration shown in Table 5.3. Since the NTCIR-12 MathIR Wikipedia task consider a Wikipedia article as the retrieval unit, we set up two databases to store math-level and document-level information. In addition, we apply math enrichment to each math expression found in the corpus, by splitting the expression on its top-level (in)equality relation symbols, as de-

scribed in Chapter 5.3.1. We then index the original and the split subexpressions into our database. Furthermore, since the output of the splitting is in the Presentation MathML format, we encode only the presentation markup of the math expressions (and subexpressions).

Table 5.3: The index system storing Wikipedia dataset for math entity linking

Index Name	**Field**
Math	p_opaths_op
	p_opaths_arg
	p_upaths_op
	p_upaths_arg
	p_sisters
	p_stree
	p_mtrick
	p_sigure
	contexts
	descriptions
	noun_phrases
	contexts_depgraph
	descriptions_depgraph
	noun_phrases_depgraph
Document	title
	abstract
	keywords
	descriptions
	noun_phrases
	all_words

We utilize 20 topics released by the NTCIR-12 MathIR Wikipedia task, each of which includes a math expression (in MathML format) and the Wikipedia article that contains the query math expression. The latter information was provided only for the assessor. Figure 5.2 shows an example of a topic. After the evaluation, this task also released the result of a pool assessment; on average, there were 67 retrieval units (i.e., math expressions) per topic, manually assessed with a relevancy score 0-4. A relevancy score of 0 denotes non-relevance, 1–2 partial relevance, and 3–4 high relevance. For each $topic_k = (m_k, t_k)$ that contains math expression m_k and the source Wikipedia article t_k, we manually annotate each retrieval unit v assessed for the $topic_k$ as:

- Matching: v has both presentation and meaning similar to m_k, and/or
- Representative: v is an important math expression contained in Wikipedia article t and is representative of its title.

Once we annotated the assessed retrieval units of $topic_k$, we create a set MR_k of retrieval units that are annotated as matching and representative, a set MN_k of

(NTCIR12-MathWikiFormula-20) Correlation_and_dependence

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Query Source: Correlation_and_dependence:1

Figure 5.2: An example of a topic from the NTCIR-12 MathIR Wikipedia task

matching-but-nonrepresentative retrieval units, and a set NN_k of nonmatching and nonrepresentative retrieval units. Subsequently, for each $topic_k$, we construct a set of math mentions $M_k = MR_k \cup MN_k$. Our training data provides a list $POS_{k,i}$ of correct links and a list $NEG_{k,i}$ of incorrect links for each math mention $m_{k,i} \in M_k$. The link candidates $T_{k,i}$ for each mention $m_{k,i}$ are defined as $T_{k,i} = POS_{k,i} \cup NEG_{k,i}$.

$$\begin{aligned} POS_{k,i} &= \{v_j | v_j \in MR_k \cup \{m_k\} \wedge title(t_{k,i}) \neq title(v_j)\} \\ NEG_{k,i} &= \{v_j | v_j \in MN_k \cup NN_k \wedge title(t_{k,i}) \neq title(v_j)\} \end{aligned} \tag{5.8}$$

The notation $title(v_j)$ represents the title of Wikipedia article from which the math expression v_j comes.

From the initial 20 topics, we utilize 16 topics for which $|MR_k| + |MN_k| > 0$. From these 16 topics, we generate 241 math mentions ($|MR_k| + |MN_k|$).

5.6.2 Experimental Design

In the experiment, we investigate the performance of several MEL approaches:

1. MIR (baseline)

$$t_i = title(\arg\max_{v'}(sim_m(m_i, v') + sim_d(m_i, v'))) \tag{5.9}$$

2. w_v^1-weighted MIR

$$t_i = title(\arg\max_{v'}(w_{v'}^1 * sim_m(m_i, v') + sim_d(m_i, v'))) \tag{5.10}$$

3. v_v^2-weighted MIR

$$t_i = title(\arg\max_{v'}(w_{v'}^2 * sim_m(m_i, v') + sim_d(m_i, v'))) \tag{5.11}$$

4. Learning to Rank with MIR features: math-level and document-level similarities

5. Learning to Rank with three features: math-level similarity, document-level similarity, and math importance

6. Learning to Rank with all features (proposed approach): math-level similarity, document-level similarity, math importance, and matching location

In the learning to rank approach, we utilized RankLib (The Lemur Project, 2015) to train a LambdaMART model using NDCG@1 as the metric to optimize on the training data. This metric is defined as:

$$\begin{aligned} NDCG@n &= \frac{DCG@n}{IDCG@n} \\ DCG@n &= \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)} \end{aligned} \tag{5.12}$$

where p is the number of candidates, rel_i is the graded relevance of the result at position i, and ideal DCG@n (IDCG@n) is the maximum possible DCG until position p.

The performance of the LambdaMART model is is evaluated using 10-fold cross validation. For each model, we report the highest performance under a single combination of hyperparameters. The hyperparameter space involves the number of trees (5000, 1000), the number of leaves for each tree (8, 10, 16), the learning rate (0.001, 0.01, 0.1), and minimum leaf support (1, 10, 100, 1000). To evaluate the disambiguation performance, we used trec_eval (NIST, 2009) to report the precision-at-1 (P@1) metric.

We utilized our final math search system developed in Chapter 4, as a baseline. The second and third approaches investigate whether our proposed importance feature is effective in improving the disambiguation performance of the MIR system. These first three approaches are considered to be unsupervised method. The final technique is our proposed supervised learning approach. To further evaluate our two proposed features, we compare the performance this final technique to the other learning approaches with less features.

5.6.3 Experimental Results and Discussion

The disambiguation performance of each approach is shown in Table 5.4. The results show that a straightforward application of MIR is ineffective for MEL, which delivers merely 6.22% precision. Interestingly, we found that by simply

Table 5.4: Disambiguation performance.

Methods	Precision (%)
Unsupervised approaches	
MIR (baseline)	6.22
w^1-weighted MIR	31.12
w^2-weighted MIR	54.77
Supervised approaches	
Learning to Rank – MIR feats.	52.85
Learning to Rank – 3 feats.	82.16
Learning to Rank – all feats. (proposed)*	83.40

weighting the math-level similarity of the MIR approach, we can improve its precision significantly. Importance feature from unweighted and weighted dependency graph increases the precision to 31.22% and 54.77%, respectively. This demonstrates that our importance feature can detect important or representative math expressions in Wikipedia articles. This finding agrees with our assumption that important math expressions often appear in the early sections of articles and are displayed on their own lines.

The use of supervised learning technique obviously gives better precision. By using only two MIR features, we already obtain 52.85% precision. Furthermore, the LambdaMART model with three features (two MIR features and importance score) significantly outperforms the previous methods. The final approach however, only offers 1.51% precision improvement relative to the model with three features.

Even though the use of matching location as a individual feature in supervised approach only gives a marginal improvement, its use for generating weighted dependency graph has an important impact. The results from the unsupervised approaches (w^1 v. w^2-weighted MIR) shows that taking into account the similarity between math expressions can produce more reliable importance scores. This shows that our matching location feature has the potential to disambiguate link candidates. One downside of this feature is that when two split subexpressions (v'_1 and v'_2) are the same numeric literal, this feature will have high value. This may cause two math expressions with different meanings to be marked as similar. For instance, this feature will tell that an algebraic equation $x^5 - 3x + 1 = 0$ have a similar meaning to the Dirac equation $(\hbar(\frac{\gamma_0}{c}\partial_t + \sum_{k=1}^{3}\gamma_k\partial_k) + imc)\Phi(t,x) = 0$, because both expressions have the same split subexpression, that is the numeric literal "0". To overcome this issue in our next MEL system, we plan to disregard the comparison between two split subexpressions that denote numeric literals or

simple symbols.

Table 5.5: MEL output examples.

No.	Math mention and the resulting link
	Correct Links
1	Mention: $\cos C = -\cos A \cos B + \sin A \sin B \cos c$ Source: Spherical_trigonometry Link: Spherical_law_of_cosines Match: $\cos(A) = -\cos(B)\cos(C) + \sin(B)\sin(C)\cos(a)$
2	Mention: $r = \frac{\sum_{i=1}^{N}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{N}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{N}(Y_i-\bar{Y})^2}}$ Source: Fisher_transformation Link: Correlation_and_dependence Match: $r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)s_x s_y} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2 \sum_{i=1}^{n}(y_i-\bar{y})^2}}$
3	Mention: $\frac{(2N)!}{k!(2N-k)!}p^k q^{2N-k}$ Source: Genetic_drift Link: Poisson_limit_theorem Match: $\frac{(n)!}{(n-k)!k!}p^k(1-p)^{n-k} \to e^{-\lambda}\frac{\lambda^k}{k!}$
	Incorrect Links
4	Mention: $ax^2 + bx + c = 0$ Source: Algebraic_solution Link: Bring_radical Match: $x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 = 0$
5	Mention: $as^2 + bs + c = 0$ Source: Huzita-Hatori_axioms Link: Line-sphere_intersection Match: $ad^2 + bd + c = 0$
6	Mention: $\begin{bmatrix} V_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ V_2 \end{bmatrix}$ Source: Two-port_network Link: Invariant_subspace Match: $T = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}$

Table 5.5 shows examples returned by our method. Our system links mention #1 to a correct Wikipedia article, which contains a math expression that has the same structure as the mention. It also returns correct links for mentions 2 and 3, although the matching math expressions do not perfectly match the mentions.

Our system can address this case because of the math enrichment results that calculate math-level similarity. We found the result for mention 3 to be quite interesting. The matching math expression is a partial match to the mention and the body of the matched article does not contain most of the textual information of the math mention (i.e., only the term "probability" is shared). Our system detects that the article containing this matching expression is the correct link based on two factors:

- The use of math enrichment boosts the math-level similarity between the mention and the math expression in the knowledge base, indicating to the system that both expressions have a similar structure.
- The matching expression appears on its own line early in the article. This gives the expression a high importance score.

Our method, however, returns incorrect links for mentions 4–6. Mention 4 is a quadratic equation, but the returned link is about the Bring radical, the unique real root of polynomial x^5+x+a. This happened because there is a partial match between the mention and the returned expression, and the description "quadratic equation" appears repeatedly in the returned Wikipedia article. Mention 5 perfectly matches the returned math expression and both represent the same concept (a quadratic equation). However, the article "Line-sphere_intersection" is not judged to be a correct link, because the returned expression is an insignificant entity in the article. For mention 6, the returned math expression has a structure similar to that of the math mention, but we can easily assess that this is an incorrect link, as these two math expressions represent different concepts, i.e., hybrid parameters in electrical circuits (the math mention) and linear mapping (the returned expression). These negative results suggest that we need to incorporate global coherence methods into our next MEL system to maximize the agreement between candidate entities.

5.7 Conclusion

In this study, we addressed the challenge of math entity linking (MEL). We found out that a straightforward application of a math search system is inadequate. We then proposed a learning-based MEL system that utilizes features, such as math-level similarity, document-level similarity, math importance, and matching location. Evaluation of the proposed system show that it achieved 83.40% precision, outperforming the straightforward application of a math search system (6.22%).

More importantly, we found that our math importance feature can detect important math expressions in Wikipedia articles. Furthermore, this confirms our assumption that important math expressions often appear near the beginning of articles and are displayed on their own lines.

A possible future work is to enhance the matching location feature. Instead of applying exact matching here, we can consider unification to find the matching subtrees for each given math expression. However, since unification is an expensive operation, we need to be careful with how many times it is applied for disambiguating each mention. Another suggestion is to develop a module to detect NIL mentions, i.e., mentions without corresponding Wikipedia title.

Chapter 6

Discussion and Future Works

In this thesis, we developed a mathematical information access (MIA) system that is composed from math search and math entity linking (MEL) modules. For the first module, we proposed the extraction of multiple types of textual information and the use of dependency relationships between math expressions. We then implemented and examined the effectiveness of these two aspects in a math search system. Later, for the purpose of formula browsing in MIA, we introduced a challenge of linking math expressions in documents to their corresponding Wikipedia articles.

In this chapter, we first discuss the limitations of our math search and MEL modules. We also discuss the roles of dependency relationships between math expressions in these two modules. Then, we describe the limitations of our MIA system and present possible future work to further advance the research in MIA.

6.1 Framework of Mathematical Search System

Math search systems are expected to allow math expressions within each document to be queried using math expressions and keywords. Therefore, there are several aspects to consider during developing a math search system, namely extraction of textual information, encoding and indexing of math expressions and their textual information, and the scoring function.

The encoding of math expressions is not our focus in this thesis. The majority of current math search systems, even though they have their own encoding implementations, the main idea is rather similar. Some search systems encode math expressions by extracting the placement of symbols in the math expressions, and some of the others extracting and generalizing the substructures of math expressions. Our path-based and hash-based (or subtree-based) techniques are in the former and latter categories, respectively. Unification, which brings positive effect to our search system, also had been used in substitution tree based

math search system.

For each math expression, we extracted and indexed several types of textual information, which can be categorized into three different levels: math, paragraph, and document. First, we extracted textual information describing each math expression. For this category, we considered words within the same sentence as the expression and also textual descriptions which are precisely define the expression. Next, we extracted two categories of textual information, i.e. the one that is shared by math expressions appearing the same paragraph and the one by expressions in the same document. In this dissertation, while we have performed several steps to obtain math-level textual information, the processing steps applied to obtain paragraph- and document-level textual information is however minimal. We suggest that there may not be much information we can extract from a paragraph, but applying additional information extraction steps to the documents can be useful for searching. For instance, the category of the documents (e.g. technical or not) and the topic of the documents (e.g. related scientific fields). In addition, we can also extract the relationships between documents. In the case of storing math expressions from scientific articles, we may use the bibliometric information of the documents. Meanwhile, for web-based articles (e.g. Wikipedia), we can exploit the links among them. This will enable us to retrieve math expressions that appear in the highly cited or popular documents.

Another limitation is the bag-of-words representation we used to store the textual information. We can further enrich the representation of each indexed math expression by constructing its vector representation using word2vec (Mikolov et al., 2013) (i.e. becoming math2vec) and/or Latent Dirichlet Allocation (Blei et al., 2003). We also found out that the architecture of our search system is not yet reliable. The query time of our search system is too long, but sharding the database should solve this issue.

In regard to the scoring and ranking in our search system, we pursued a technique that is simple to implement, so that it would not add complexity to our math search pipeline. We applied cold-start weighting schema obtained from multiple linear regression. To implement this weighting, we modified the queries sent to our database to take into account the cold-start weight information. Despite its simplicity, we did not get a positive result from this scenario. We consider multicollinearity as the main issue here, because we attempted to fit a regression model using all variables (i.e. fields in our databases) that might be correlated to each other. A possible solution here is to reduce the number of the scores returned in searching process. For instance, we can group the scores from several

fields together, so that in addition to eliminate the multicollinearity issue, we can also get scores that are easy to interpret, such as math similarity score, text similarity in math-level database, paragraph similarity, and document similarity. Another solution is the implementation of other learning to rank algorithms, such as SVM^{rank} (Joachims, 2006) and ListNet (Cao et al., 2007), to handle multiple fields. The implementation of learning to rank technique as a post-processing module will also open the possibilities to rerank initial ranked lists using features other than the initial scores, such as the importance of math expression in the containing document, the topic of the document, the similarity between math expressions in the initial ranked list (a relevant math expression might be the one that appears several times in the top-N of the ranked list), and whether the query and the retrieved math expression are unifiable.

Unification technique, which is the final step in our math search system, is capable of improving the search precision, but prolongs the query time. Since unification can be used to rerank the initial ranked lists, we should carefully determine the number of retrieval units in the initial lists.

6.2 Linking Mathematical Expressions to Wikipedia

We introduced a challenge and a learning-based system to link math expressions found in documents to their corresponding Wikipedia articles. We proposed a technique to estimate the importance of math expressions in Wikipedia articles, since the matching math expressions have to be important in the containing articles. There are three factors we took into account in this technique: (i) the location of the math expression in the document, (ii) the display of the math expression, and (iii) how similar the math expression to the other math expressions in the document. We encode these three types of information into dependency graph of math expressions, then apply Personalized PageRank algorithm to the graph.

Our experimental results showed that this importance score works well in identifying correct links. However, the factors used to compute the importance score requires several assumption. First, it assumes that the later a math expression appears in an article, the less important it is. This assumption is reasonable if an article explaining a certain concept puts majority of the definition of the concept in the early sections, and put only extra information, such as examples, applications, and derivation of the main math expressions, in the later sections. However, some articles in Wikipedia have different formats. They explain a general topic, and further describe more specific concepts in each section. For

instance, the article titled "Information_theory" contains several concepts, such as entropy, mutual information, and coding theory, in different sections. Since this article can be considered as a correct link to any math expressions from these concepts, these concepts have the same importance in this article. Thus, the order of the sections in which these concepts appear should not be taken into account for computing the importance score. This issue is a challenging task to tackle in the future, since we need to initially predict the role of each section in each Wikipedia article. This role will be useful as a disambiguation feature.

Furthermore, regarding how math expressions are displayed, the math expressions in equation environment are often important, but math expressions appearing inline are not necessarily insignificant. Additional information that can be used to tell whether math expressions are important or not is to recognize whether expressions are math formulae (i.e. holding (in)equalities) or not. In addition, we can treat a simple symbol in a Wikipedia article as an important concept if this article specifies a math formula to define it.

Despite the promising performance of our MEL module, we have not considered the effect of NIL link. In this dissertation, we assume that all math expressions have their corresponding Wikipedia articles. This assumption was also frequently used by wikification (or entity linking) work that focused on developing disambiguation algorithms. In the actual run, this assumption however, may not hold. We consider this aspect as the main limitation of our MEL module. The simplest way to solve this in the future is to implement a classifier to tell us whether the disambiguation result of a math mention is the correct link or not. For this classifier, we can use the confidence distribution of the disambiguation candidates, link probability (Mihalcea and Csomai, 2007), and generality of each disambiguation result.

An interesting extension of this work is to leverage the wikification result of textual concepts. We can leverage the disambiguation results from current wikification (or entity linking) system, by considering them during the computation of the global coherence of a disambiguation candidate of a math expression. Furthermore, we can jointly disambiguate textual concepts and math expressions in documents.

In addition, in this dissertation, we defined that the disambiguation result is a Wikipedia article that contains math expression(s) similar to the math mention and also mainly explain the same concept as the mention. For future direction, we may relax the first rule, i.e. having a matching concept is more important than a matching math expression, since this is more suitable to the definition of

wikification or entity linking.

6.3 Utilizing Dependency Relationships between Mathematical Expressions in Mathematical Information Access

Both our math search and MEL modules gain benefits from dependency graph. Our math search system uses it to enrich the indexed textual information. Meanwhile, our MEL module utilizes it to derive the importance score of math expressions in documents. As a consequence, we obtained satisfactory experimental results from these modules. This reflects that to semantically disambiguate math expressions, we need to take into account the textual meanings of those expressions and their constituent symbols. The current condition in MIA, where has not been able yet to automatically disambiguate math expressions from only their presentations, may also back the importance of this textual enrichment.

Another interesting application of dependency graph is to capture relationships between math expressions from multiple documents. While attempting to do it in an arbitrary collection of technical documents might be exceptionally difficult, especially due to variations in mathematical notation across scientific fields, it is interesting to examine its possibility in a set of documents from closely related topics. Furthermore, once we can capture the mathematical concept represented by each math expression reliably (Pagel and Schubotz, 2014; Schubotz et al., 2016), we can transform a dependency graph of math expressions into a knowledge graph of math concepts.

In this dissertation, we proposed a heuristic algorithm based on string matching and extensive normalization of math expressions. Our method is based on the assumption that two math expressions having the same (base) representation or the same left-hand side subexpressions is equivalent to sharing the same meaning. This assumption is reasonable if each dependency graph covers only a single document. However, to develop a dependency graph that covers multiple documents, this assumption cannot be used, as different documents may use different math symbols to represent the same math concept and the same math symbol to represent different math concepts. As a workaround, we can consider extracting a dependency graph by exploiting more information related to each math expression, for example, by exploiting both the structures of the math expressions and the words surrounding each math expression.

Chapter 7

Conclusion

The main challenge in developing the math search and math entity linking (MEL) modules of a math information access (MIA) system is the ambiguity inherent in math expressions. Most of the previous work on math search system focused on establishing index systems, and have not yet fully exploited the text in the documents. Each of these systems utilized only a certain type of textual information. However, each type of textual information has its own strength and weakness.

Chapter 3 describes the framework of our proposed math search system. This framework includes the preprocessing step applied to each math expression, the indexing step, and the post-processing module. In the preprocessing step, we first encode math expressions, so that we can extract their structures. We applied two encoding techniques, i.e. path- and hash (or subtree)-based techniques, to both the presentation and content markup of math expressions. Furthermore, since we cannot acquire the semantics of math expressions only from their token elements and structures, we extract multiple types of textual information to represent each math expression. This is the first unique feature of our math search system. We have three different levels of detail of the textual information, namely document-, paragraph-, and math-level textual information. The paragraph- and document-level information is used to capture all textual concepts that appears in the paragraph and document containing math expression. These types of information support our search system to have a good recall. On the other hand, the math-level textual information specifically describes the meaning of each math expression. Extracting and indexing this information allows our math search to have a good precision. Once we indexed both the encoded representation and the textual information of math expressions, we setup the scoring component of the search system to ensure that there is no scores from certain fields improperly dominate the final score during searching. In addition, different database fields may have different importance, i.e., certain fields may be more predictive of the

relevance of math expressions. To meet this requirement, we implement score normalization, which is surprisingly often overlooked by other search systems, and a weighting scheme. While the score normalization successfully prevents bias toward certain fields, the weighting scheme does not perform well. We suggest that the underperforming weighting scheme, which is obtained by multiple linear regression, is caused either by development set and test are constructed by assessors using different relevancy criteria, or by the multicollinearity issue in the regression model. This latter issues arise because the introduction of too many fields (i.e. variables in the regression model) that are correlated to each other. Finally, we implement unification as post-processing module in our search system. Despite its time-costly operation, it delivers precision improvement to our system. Our two key features, the use of multiple types of textual information and score normalization, allow our system to outperform majority of the participating systems in the NTCIR-12 MathIR task.

Chapter 4 discuss the extraction and impact of dependency graph of math expressions in our math search system. We first describe our proposed heuristic method to extract dependency graphs. We evaluate its effectiveness using manually annotated data and show that the proposed method delivers an accuracy of .8674 while the baseline (unification) method achieves an accuracy of .7140. We then evaluate the effectiveness of the dependency graph in enriching textual information related to math expressions and in improving the retrieval results of math search system. The use of dependency graph to further enrich the math-level textual information is another unique feature of our math search system. We can interpret the textual enrichment by dependency graph as the global context of math expressions. Prior to the use of dependency graph, our search system attempts to retrieve only math expressions that have meanings similar to the user queries. However, dependency graph allows us to also retrieve other math expressions expressing concepts that are related to the user queries. Our experimental result shows that the use of dependency graph in the math search system delivers 12.60% precision improvement. Furthermore, in the recent NTCIR-12 MathIR task, we record that dependency graph significantly improves precision of our search module by 24.52%. Overall, the best setting of our math search system that incorporates this graph achieves precision-at-5 of .2828 and precision-at-20 of .1586 in relevant judgment, as well as .5448 and .4897 in partially relevant judgment. On the other hand, the next best system delivered .2276 and .1362 together with .5517 and .4000 in relevant and partially relevant judgments, respectively.

In Chapter 5, we introduce a challenge and a learning-based system to link math expressions found in documents to their corresponding Wikipedia articles. We first approached this problem using our math search system. However, the result was unsatisfactory. We noticed that even though both math search and this linking problems attempt to find math expressions semantically similar to the query (or math mention), there is a unique challenge in this linking problem. The matching math expressions need to be not only semantically similar to the mention, but also important in the containing documents. To solve this unique challenge, we introduced a technique to estimate the importance of math expressions in Wikipedia articles. There are three factors we took into account in this technique: (i) the location of the math expression in the document, (ii) the display of the math expression, and (iii) how similar the math expression to the other math expressions in the document. We encode these three types of information into dependency graph of math expressions, then apply Personalized PageRank algorithm to the graph. Our experimental results showed that this feature is crucial to solve this challenge. It allowed our unsupervised approach to have a precision of 54.77% from previously 6.22%. The use of supervised approach allowed us to have an even higher precision.

The cornerstone of our work was the dependency relationships between math expressions. Both our math search module and math entity linking modules gain benefits from these dependency relationships. This reflects that such dependency relationships have potential to semantically disambiguate math expressions.

This work was our contribution towards enabling mathematics-aware information retrieval and the disambiguation of mathematical knowledge in documents. We hope that digital library practices in the near future integrate mathematical information access modules into their implementation.

References

Adeel, M., Cheung, H. S., and Khiyal, A. H. (2008). Math GO! prototype of a content based mathematical formula search engine. *Journal of Theoretical and Applied Information Technology*, 4:1002–1012.

Agrawal, R., Gollapudi, S., Kannan, A., and Kenthapadi, K. (2012). Data mining for improving textbooks. *SIGKDD Explorations Newsletter*, 13(2):7–19.

Aizawa, A., Kohlhase, M., and Ounis, I. (2013). NTCIR-10 math pilot task overview. In *Proc. of the 10th NTCIR Conference*.

Aizawa, A., Kohlhase, M., Ounis, I., and Schubotz, M. (2014). NTCIR-11 math-2 task overview. In *Proc. of the 11th NTCIR Conference*.

Alhelbawy, A. and Gaizauskas, R. (2014). Graph ranking for collective named entity disambiguation. In *Proc. of the 52nd ACL*.

Anastacio, I., Martins, B., and Calado, P. (2011). Sueprvised learning for linking named entities to knowledge base entries. In *Proc. of the TAC2011*.

Asperti, A. and Selmi, M. (2004). Efficient retrieval of mathematical statements. In *Proc. of the 3rd International Conference on Mathematical Knowledged Management*, pages 17–31.

Ausbrooks, R., Buswell, S., Carlisle, D., Chavchanidze, G., Dalmas, S., Devitt, S., Diaz, A., Dooley, S., Hunter, R., Ion, P., Kohlhase, M., Lazrek, A., Libbrecht, P., Miller, B., Miner, R., Rowley, C., Sargent, M., Smith, B., Soiffer, N., Sutor, R., and Watt, S. (2014). Mathematical markup language (MathML) version 3.0 2nd edition. *W3C Recommendation*.

Baker, J. B., Sexton, A. P., and Sorge, V. (2012). MaxTract: converting PDF to LATEX, MathML and text. In *Proc. of the 11th CICM*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(4–5):993—1022.

Bunescu, R. and Paşca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *Proc. of the 11th EACL*.

Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82.

Buswell, S., Caprotti, O., Carlisle, D., Dewar, M., Gaetano, M., and Kohlhase, M. (2004). The openmath standard. Technical report, The Open Math Society.

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

Ceccarelli, D., Lucchese, C., Orlando, S., Perego, R., and Trani, S. (2013). Learning relatedness measures for entity linking. In *Proc. of the 22nd ACM CIKM*.

Chang, M., Srikumar, V., Goldwasser, D., and Roth, D. (2010). Structured output learning with indirect supervision. In *Proc. of the ICML*.

Chen, Z. and Ji, H. (2011). Collaborative ranking: A case study on entity linking. In *Proc. of EMNLP2011*.

Cheng, X. and Roth, D. (2013). Relational inference for wikification. In *Proc. of EMNLP*.

Cilibrasi, R. and Vitanyi, P. M. B. (2007). The google similarity distance. *IEEE Trans. Knowledge and Data Engineering*, 19(3):370–383.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on Wikipedia data. In *Proc. of the 2007 Joint Conference of EMNLP-CoNLL*.

Davila, K., Zanibbi, R., Kane, A., and Tompa, F. W. (2016). Tangent-3 at the NTCIR-12 MathIR task. In *Proc. of the 12th NTCIR Conference*.

Davis, A., Veloso, A., da Silva, A. S., W. Meira, J., and Laender, A. H. F. (2012). Named entity disambiguation in streaming data. In *Proc. of the 50th ACL*.

de Pablo-Sanchez, C., Perea, J., and Martinez, P. (2010). Combining similarities with regression based classifiers for entity linking at TAC 2010. In *Proc. of TAC2010 Workshop*.

Dredze, M., McNamee, P., Rao, D., Gerber, A., and Finin, T. (2010). Entity disambiguation for knowledge base population. In *Proc. of the 23rd COLING*.

Erkan, G. and Radev, D. R. (2004). LexRank: graph-based lexical centrality a salience in text summarizatoin. *Journal of Artificial Intelligence Research*, 22(1):457–479.

Fernandez, N., Fisteus, J., Sanchez, L., and Martin, E. (2010). Webtlab; a cooccurence-based approach to KBP 2010 entitiy-linking task. In *Proc. of the 3rd TAC*.

Ferragina, P. and Scaiella, U. (2010). Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proc. of the 19th ACM CIKM*.

Finkel, J. R., Grenager, T., and Manning, C. D. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of the 43rd ACL*.

Florian, R., Jing, H., Kambhatla, N., and Zitouni, I. (2006). Factoring complex models: A case study in mention detection. In *Proc. of the 44th ACL*.

G. Demartini, D. E. D. and Cudré-Mauroux, P. (2012). Zencrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proc. of the 21st WWW*.

Gao, L., Wang, Y., Hao, L., and Tang, Z. (2014). ICST math retrieval system for NTCIR-11 math-2 task. In *Proc. of the 11th NTCIR Conference*.

Gao, L., Yuan, K., Wang, Y., Jiang, Z., and Tang, Z. (2016). The math retrieval system of ICST for NTCIR-12 MathIR task. In *Proc. of the 12th NTCIR Conference*.

Gonzalez, E., Rodriguez, H., Turmo, J., Comas, P. R., Naderi, A., Ageno, A., Sapena, E., Vila, M., and Marti, M. A. (2012). The TALP participation at TAC-KBP 2012. In *Proc. of the TAC2012*.

Graf, P. (1996). *Term Indexing*, volume 1053 of *LNCS*. Springer Verlag.

Gray, J. (2007). ASCIIMathML: Now everyone can type MathML. *MSOR Connections*, 7(3).

Grigore, M., Wolska, M., and Kohlhase, M. (2009). Towards context-based disambiguation of mathematical expressions. In *Proc. of the Joint conference of ASCM-09 and MACIS-09*, volume 22 of *Math-for-Industry Lecture Notes Series*, pages 262–271.

Guidi, F. and Sacerdoti Coen, C. (2015). A survey on retrieval of mathematical knowledge. In *Proc. of the 8th Conference on Intelligent Computer Mathematics*.

Guo, S., Chang, M.-W., and Kiciman, E. (2013). To link or not to link? a study on end-to-end tweet entity linking. In *Proc. of the NAACL-HLT*.

Guo, Z. and Barbosa, D. (2014). Robust entity linking via random walks. In *Proc. of the 23rd ACM CIKM*.

Hachey, B., Radford, W., and Curran, J. R. (2011). Graph-based named entity linking with Wikipedia. In *Proc. of the 12th WISE*.

Hachey, B., Radford, W., Nothman, J., Honnibal, M., and Curran, J. R. (2013). Evaluating entity linking with Wikipedia. *Artificial Intelligence*, 194(0):130–150.

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

Han, X. and Sun, L. (2011). A generative entity-mention model for linking entities with knowledge base. In *Proc. of the 49th ACL-HLT*.

Han, X., Sun, L., and Zhao, J. (2011). Collective entity linking in web text: A graph-based method. In *Proc. of the 34th ACM SIGIR*.

Hashimoto, H., Hijikata, Y., and Nishida, S. (2007). A survey of index formats for the search of MathML objects. *IPSJ SIG Technical Reports, DBS-142, FI-87*, pages 55–59.

Haveliwala, T. H. (2003). Topic-sensitive PageRank: A context-sensitive ranking agorithm for web search. *IEEE Transaction on Knowledge and Data Engineering*, 15(4):784–796.

Hijikata, Y., Hashimoto, H., and Nishida, S. (2007). An investigation of index formats for the search of MathML objects. In *Proc. of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, pages 244–248.

Hu, X., Gao, L., Lin, X., Tang, Z., Lin, X., and Baker, J. B. (2013). WikiMirs: A mathematical information retrieval system for wikipedia. In *Proc. of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*.

Huang, H., Cao, Y., Huang, X., Ji, H., and Lin, C.-Y. (2014). Collective tweet wikification based on semi-supervised graph regularization. In *Proc. of the 52nd ACL*.

InftyProject (2015). Infty reader. `http://www.sciaccess.net/en/InftyReader/`.

Jinha, A. E. (2010). Article 50 million: An estimate of the number of scholarly articles in existence. *Learned Publishing*, 23(3):258–263.

Joachims, T. (2006). Training linear svms in linear time. In *Proc. of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

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

Kamali, S. and Tompa, F. W. (2013b). Structural similarity search for mathematics retrieval. In *Proc. of Conferences on Intelligent Computer Mathematics*.

Knuth, D. E. (1984). *The TEXbook*. Addison Wesley.

Kohlhase, A. and Kohlhase, M. (2007). Reexamining the MKM value proposition: From math web search to math web research. In *Calculemus '07 / MKM '07: Proc. of the 14th symposium on Towards Mechanized Mathematical Assistants, LNAI*, volume 4573, pages 313–326, Berlim, Heidelberg.

Kohlhase, M. and Ginev, D. (2006–2016). arXMLiv. `http://arxmliv.kwarc.info/`.

Kohlhase, M., Matican, B. A., and Prodescu, C.-C. (2012). MathWebSearch 0.5 – scaling an open formula search engine. In *Proc. of Conferences on Intelligent Computer Mathematics 2012, LNAI*, volume 7362, pages 342–357.

Kohlhase, M. and Sucan, I. (2006). A search engine for mathematical formulae. In *Proc. of the 8th International Conference AISC, LNAI*, volume 4120, pages 241–253. Springer.

Kristianto, G. Y., Nghiem, M.-Q., Inui, N., Topić, G., and Aizawa, A. (2012a). Annotating mathematical expression definitions for automatic detection. In *Proc. of the CICM Workshop "Mathematics Information Retrieval"*.

Kristianto, G. Y., Topić, G., and Aizawa, A. (2014a). Exploiting textual descriptions and dependency graph for searching mathematical expressions in scientific

papers. In *Proc. of the 9th International Conference on Digital Information Management*.

Kristianto, G. Y., Topić, G., and Aizawa, A. (2014b). Extracting textual descriptions of mathematical expressions in scientific papers. In *Proc. of the 3rd International Workshop on Mining Scientific Publications of the International Conference on Digital Libraries 2014*.

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

Kristianto, G. Y., Topić, G., Nghiem, M.-Q., and Aizawa, A. (2012b). Annotating scientific papers for mathematical formulae search. In *Proc. of the 5th Workshop on Exploiting Semantic Annotations in Information Retrieval of The 21st ACM CIKM*.

Lamport, L. (1994). *LaTeX: A Document Preparation System, 2/e*. Addison Wesley.

Larsen, P. O. and von Ins, M. (2010). The rate of growth in scientiif publication and the decline in converage provided by science citation index. *Scientometrics*, 84(3):575–603.

Lehmann, J., Monahan, S., Nezda, L., Jung, A., and Shi, Y. (2010). LCC approaches to knowledge base population at tac 2010. In *Proc. of the TAC2010 Workshop*.

Li, Q. and Ji, H. (2014). Incremental joint extraction of entity mentions and relations. In *Proc. of the 52nd ACL*.

Li, Q., Li, H., Ji, H., Wang, W., Zheng, J., and Huang, F. (2012). Joint bilingual name tagging for parallel corpora. In *Proc. of the 21st ACM International Conference on Information and Knowledge Management*.

Libbrecht, P. and Melis, E. (2006). Methods to acess and retrieve mathematical content in activemath. In *Mathematical Software - ICMS 2006, Second International Congress on Mathematical Software*, pages 331–342.

Lin, X., Gao, L., Hu, X., Tang, Z., Xiao, Y., and Liu, X. (2014). A mathematics retrieval system for formulae in layout presentation. In *Proc. of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

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

Líška, M. (2013). Evaluation of mathematics retrieval. diploma thesis, Masaryk University, Brno, Faculty of Informatics.

Luo, G., Huang, X., Lin, C.-Y., and Nie, Z. (2015). Joint named entity recognition and disambiguation. In *Proc. of the EMNLP*.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIR Press.

Medelyan, O., Witten, I. H., and Milne, D. (2008). Topic indexing with Wikipedia. In *Proc. of the AAAI 2008 Workshop on Wikipedia and Artificial Intelligence*.

Meij, E., Weerkamp, W., and de Rijke, M. (2012). Adding semantics to microblog posts. In *Proc. of the 5th WSDM*.

Mendes, P. N., Jakob, M., Garcia-Silva, A., and Bizer, C. (2011). Dbpedia spotlight: Shedding light on the web of documents. In *Proc. of the 7th I-SEMANTICS*.

Mihalcea, R. and Csomai, A. (2007). Wikify!: linking documents to encyclopedic knowledge. In *Proc. of the 16th ACM CIKM*.

Mihalcea, R. and Tarau, P. (2004). TextRank: bringing order into texts. In *Proc. of EMNLP2004*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Miller, B. (2016). LaTeXML: A LaTeX to XML/HTML/MathML converter. `http://dlmf.nist.gov/LaTeXML/`.

Miller, B. R. and Youssef, A. (2003). Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 38(1–3):121–136.

Milne, D. and Witten, I. H. (2008). Learning to link with Wikipedia. In *Proc. of the 17th ACM CIKM*.

Miner, R. and Munavalli, R. (2007). An approach to mathematical search through query formulation and data normalization. In *Proc. of the 14th symposium on Towards Mechanized Mathematical Assistants*.

Munavalli, R. and Miner, R. (2006). MathFind: A math-aware search engine. In *Proc. of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Nghiem, M.-Q., Yokoi, K., Matsubayashi, Y., and Aizawa, A. (2010). Mining coreference relations between formulas and text using wikipedia. In *Proc. of the 2nd Workshop on International Workshop on NLP Challenges in the Information Explosion Era*, pages 69–74.

Nguyen, T. T., Chang, K., and Hui, S. C. (2012a). A math-aware search engine for math question answering system. In *Proc. of the 21st ACM International Conference on Information and Knowledge Management*.

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

Ni, Y., Xu, Q. K., Cao, F., Mass, Y., Sheinwald, D., Zhu, H. J., and Cao, S. S. (2016). Semantic documents relatedness using concept graph representation. In *Proc. of the 9th ACM WSDM*.

NIST (2009). trec_eval. `http://trec.nist.gov/trec_eval/`.

Ohashi, S., Kristianto, G. Y., Topić, G., and Aizawa, A. (2016). Efficient algorithm for math formula semantic search. *IEICE TRANSACTIONS on Information and Systems*, E99-D(4):979–988.

Pagel, R. and Schubotz, M. (2014). Mathematical Language Processing Project. In *Work in Progress track at CICM*.

Pattaniyil, N. and Zanibbi, R. (2014). 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*.

Pinto, J. M. G. and Balke, W.-T. (2015). Demystifying the semantics of relevant objects in scholarly collections: A probabilistic approach. In *Proc. of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries*.

Pinto, J. M. G., Barthel, S., and Balke, W.-T. (2014). QUALIBETA at the NTCIR-11 math 2 task: An attempt to query math collections. In *Proc. of the 11th NTCIR Conference*.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proc. of the 13th CoNLL*.

Ratinov, L., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to wikipedia. In *Proc. of the 49th ACL: Human Language Technologies*.

Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents. In *Text Mining: Applications and Theory*.

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

Růžička, M., Sojka, P., and Líška, M. (2016). Math indexer and searcher under the hood: Fine-tuning query expansion and unification strategies. In *Proc. of the 12th NTCIR Conference*.

Salton, G. and Buckley, C. (1988). Term-weighting appproaches in automatic text retrieval. *Information Processing Management*, 24(5):513–523.

Schellenberg, T. (2011). Layout-based substitution tree indexing and retrieval for mathematical expressions. Master's thesis, Rochester Institute of Technology.

Schellenberg, T., Yuan, B., and Zanibbi, R. (2012). Layout-based substitution tree indexing and retrieval for mathematical expressions. In *Proc. of the 19th Document Recognition and Retrieval*.

Schubotz, M., Grigorev, A., Leich, M., Cohl, H. S., Meuschke, N., Gipp, B., Youssef, A. S., and Markl, V. (2016). Semantification of identifiers in mathematics for better math information retrieval. In *Proc. of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Schubotz, M., Leich, M., and Markl, V. (2013). Querying large collections of mathematical publications. In *Proc. of the 10th NTCIR Conference*.

Schubotz, M., Youssef, A., Markl, V., and Cohl, H. S. (2015). Challenges of mathematical information retrieval in the NTCIR-11 math wikipedia task. In *Proc. of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Schubotz, M., Youssef, A., Markl, V., Cohl, H. S., and Li, J. J. (2014). Evaluation of similarity-measure factors for formulae based on the NTCIR-11 math task. In *Proc. of the 11th NTCIR Conference*.

Sil, A. and Yates, A. (2013). Re-ranking for joint named-entity recognition and linking. In *Proc. of the 22nd ACM CIKM*.

Sojka, P. (2012). Exploiting semantic annotations in math information retrieval. In *Proc. of the 5th Workshop on Exploiting Semantic Annotations in Information Retrieval of The 21st ACM CIKM*, pages 15–16.

Sojka, P. and Líška, M. (2011a). The art of mathematics retrieval. In *Proc. of the 11th ACM Symposium on Document Engineering*.

Sojka, P. and Líška, M. (2011b). Indexing and searching mathematics in digital libraries - architecture, design and scalability issues. In *Proc. of Conferences on Intelligent Computer Mathematics 2011*, pages 228–243.

Stathopoulos, Y. and Teufel, S. (2015). Retrieval of research-level mathematical information needs: A test collection and technical terminology experiment. In *Proc. of the 53rd Annual Meeting of the Association for Computational LInguistics and the 7th International Joint Conference on Natural Language Processing*.

Taskar, B., Guestrin, C., and Koller, D. (2004). Max-margin markov networks. In *Proc. of NIPS*.

Thanda, A., Agarwal, A., Singla, K., Prakash, A., and Gupta, A. (2016). A document retrieval system for math queries. In *Proc. of the 12th NTCIR Conference*.

The Apache Software Foundation (2015a). Apache lucene. `http://lucene.apache.org/`.

The Apache Software Foundation (2015b). Apache solr v5.3. `http://lucene.apache.org/solr/`.

The Lemur Project (2015). RankLib. `https://sourceforge.net/p/lemur/wiki/RankLib/`.

Topić, G., Kristianto, G. Y., Nghiem, M.-Q., and Aizawa, A. (2013). The MCAT math retrieval system for NTCIR-10 Math track. In *Proc. of the 10th NTCIR Conference*.

Tscohantaridris, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Wang, Y., Gao, L., Wang, S., Tang, Z., Liu, X., and Yuan, K. (2015). WikiMirs 3.0: a hybrid mir system based on the context, structure and importance of formulae in a document. In *Proc. of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries*.

Wielemaker, J. (2015). Swi Prolog. `http://www.swi-prolog.org/`.

Wolska, M. and Grigore, M. (2010). Symbol declarations in mathemaical writing: A corpus study. In *Proc. of the CICM Workshop "Towards a Digital Mathematical Library"*, pages 119–127.

Wolska, M., Grigore, M., and Kohlhase, M. (2011). Using discourse context to interpret object-denoting mathematical expressions. In *Proc. of the CICM Workshop "Towards a Digital Mathematics Library"*, pages 85–101.

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

Wu, S., Bi, Y., and Zeng, X. (2011). The linear combination data fusion method in information retrieval. In *Proc. of the 22nd International Conference on Database and Expert Systems Applications*.

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

Yokoi, K. and Aizawa, A. (2009). An approach to similarity search for mathematical expressions using MathML. In *Proc. of the 2nd Workshop Towards a Digital Mathematics Library*, pages 27–35.

Yokoi, K., Nghiem, M.-Q., Matsubayashi, Y., and Aizawa, A. (2011). Contextual analysis of mathematical expressions for advanced mathematical search. In *Proc. of the 12th International Conference on Intelligent Text Processing and Comptational Linguistics*.

Youssef, A. (2005). Information search and retrieval of mathematical contents: Issue and methods. In *Proc. of the ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering*.

Youssef, A. (2006). Roles of math search in mathematics. In *Proc. of the 5th Mathematical Knowledge Management*, pages 2–16.

Youssef, A. (2007a). Advances in math search. In *Proc. of the 6th International Congress on Industrial and Applied Mathematics*.

Youssef, A. (2007b). Methods of relevance ranking and hit-content generation in math search. In *Proc. of the 14th Symposium on Towards Mechanized Mathematical Assistants*.

Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I., Topić, G., and Davila, K. (2016). NTCIR-12 MathIR Task Overview. In *NTCIR*. National Institute of Informatics (NII).

Zanibbi, R. and Blostein, D. (2012). Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition*, 15(4):331–357.

Zhang, Q. and Youssef, A. (2014). An approach to math-similarity search. In *Proc. of Conferences on Intelligent Computer Mathematics*.

Zhao, J., Kan, M.-Y., and Thang, Y. L. (2008). Math information retrieval: User requirements and prototype implementation. In *Proc. of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries*.

Zuo, Z., Kasneci, G., Gruetze, T., and Naumann, F. (2014). BEL: Bagging for entity linking. In *Proc. of the 25th COLING*.