NANYANG TECHNOLOGICAL UNIVERSITY

Data Mining for Mathematical Question Answering Community

MA KAI

A thesis submitted to the
Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

School of Computer Engineering

2011

Abstract

Question Answering (QA) communities such as Yahoo! Answers and Baidu Zhidao are currently very popular with millions of users. The QA communities are particularly useful for the educational domain such as mathematics. Similar to traditional QA communities, a mathematical QA community should also allow users to search, ask, answer and discover mathematical questions. However, as mathematical formulas are highly symbolic and structured, it is challenging to develop such a mathematical QA community. In this research, we aim to propose efficient and effective techniques for supporting the "search, ask, answer and discover" framework for a mathematical QA community. In particular, we focus on investigating different data mining techniques for mathematical question search, mathematical question topic classification and human expert finding. Mathematical question search will help retrieve a set of similar mathematical problems together with the answers posted by other users. Mathematical question topic classification will help recommend the possible topics of user posted questions. Human expert finding will help find a list of experts who are most likely able to answer a posted question according to their expertise.

As a result of this research, we have proposed different approaches for mathematical search, mathematical question topic classification and human expert finding based on data mining techniques. The proposed approaches and techniques have been incorporated into the Mathematical Question Answering (MathQA) Community. The performance of the proposed approaches have been evaluated and the experimental results have shown that the proposed approaches have achieved promising performance. In this thesis, the motivation, objectives, related work, and the proposed approaches and their performance evaluations will be presented. In addition, the thesis will also discuss the directions for future research work.

Acknowledgement

I am deeply indebted to my supervisor, Associate Professor Hui Siu Cheung, for his support, encouragement, and invaluable advice during the course of my Master study. Without his conscientious guidance and thoughtful criticism, this thesis would not have been completed.

I would like to thank a Ph.D student - Li Guangxia for his invaluable advices on the application of Support Vector Machines in terms of the experimental setting and key parameter adjustment. And I would also like to thank the Ph.D students - Wanglin and Li Chenliang for their careful guidance on using the typesetting and plotting software.

My gratitude also goes to those Final Year Project (FYP) students for their friendly supports and help, especially for their endless efforts on the dataset preparation for this research work.

This thesis is dedicated to my parents, whose love and unconditional support provide a constant inspiration in my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Question Answering (QA) communities [1, 2, 3, 4, 5, 6, 7, 8] such as Yahoo! Answers [9] and Baidu Zhidao [10] are very popular with millions of users. Such QA communities are highly effective to help users solve general question problems. Apart from such open domain QA communities, there are also many closed domain QA communities such as The Math Forum at Drexel University [11] and Cramster [12] aiming at specialized topical domains. The Math Forum at Drexel University is a QA community for mathematics, whereas Cramster aims for many academic subjects such as mathematics, computer science and biology. Most QA communities provide a supporting framework based on "search, ask, answer and discover". In a QA community, users can search for similar and related questions from the existing QA database. They can also browse the underlying QA database to discover any related or interesting questions. Moreover, users can post their questions online in order to get answers from other users or experts within the QA community.

The QA communities are particularly useful for the educational domain such as mathematics as demonstrated by The Math Forum at Drexel University and Cramster. Traditionally, when a student finds a mathematics problem hard to solve with his existing knowledge, he will first refer to the textbook to learn from the solutions of similar problems. If the student is still unable to solve the problem, he will then ask his peers or teachers for help. However, such traditional problem solving approach is quite ineffective as the useful resources may not always be available. With a mathematical QA community, a user can search the posted mathematical questions for finding similar questions which have been solved before.

If that is found, the user will then be able to learn from the suggested solutions of the posted questions. However, if that is not available, the user may then post his question online for other users to answer, or look for experts from the related topics for help in order to solve his question problem.

In this research, we aim to investigate the development of a QA community for mathematics. In particular, we focus on investigating data mining techniques such as clustering and classification for developing different functions for the mathematical QA community in order to support mathematical problem solving efficiently and effectively.

1.2 Mathematical Question Answering Community

A mathematical QA community aims to support problem solving for mathematical questions in the educational domain. A mathematical question is a document which comprises a problem description, formulas, graphs or/and diagrams. Figure 1.1 shows an example of a mathematical question document. Such question could be posted or answered by any users in the mathematical QA community.

The triangle XYZ has XZ $= 6$, $YZ = x$, $XY = z$ as shown below. The perimeter of triangle XYZ is 16.

1. Express z in term of x.
2. Using the cosine rule, express z^2 in term of x and $cosZ$.
3. Hence, show that $cosZ = \frac{5x-16}{3x}$.

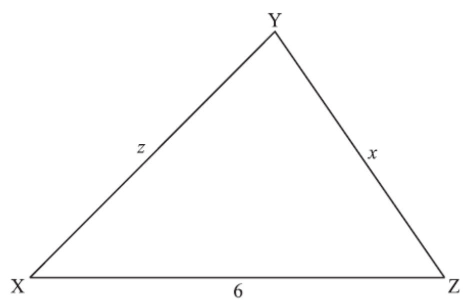


Figure 1.1: An Example Mathematical Question Document.

To develop a mathematical QA community, it is important to support the "search, ask, answer and discover" functional framework as demonstrated by existing QA communities such as Yahoo! Answers [9] and Baidu Zhidao [10]. To achieve this, we need to consider the following important technical issues: mathematical retrieval, mathematical topic classification and human expert finding.

1.2.1 Mathematical Retrieval

A mathematical QA community should support any users to search for related mathematical questions from the underlying QA database. Apart from textual problem description, a mathematical question may also contain formulas, graphs or/and diagrams. In this research, apart from text retrieval, we also investigate techniques for formula retrieval. Text retrieval is quite straightforward. However, searching formulas effectively is challenging as mathematical formulas are highly symbolic and structured. Most current mathematical formula search techniques [13, 14, 15, 16, 17] are based on the text retrieval approach. However, traditional text retrieval approaches may not be appropriate for the retrieval of formula data. In addition, a user also may enter a mathematical question query which contains both textual description and formulas. Therefore, apart from supporting formula retrieval and text retrieval separately, we also need to consider the retrieval of a mathematical question query which contains both textual description and formulas together.

1.2.2 Mathematical Topic Classification

A mathematical QA community should also support automatic question topic classification for users' posted questions. It recommends to users the possible topics for their posted questions. Classifying the posted questions appropriately can help ensure that the questions are placed in the correct topic which can then be discovered and answered subsequently. More importantly, automatic mathematical topic classification also helps the administrator to manage and organize the posted questions in topics more efficiently. Classical text classification algorithms such as Decision Trees (DTs) [18, 19, 20, 21], Support Vector Machine (SVM) [22, 23, 24, 25], Naïve Bayes (NB) [26, 27, 28, 29] and k-Nearest Neighbor (k-NN) [30, 31, 32, 33] have been applied to topic classification for different areas of applications such as email classification and news classification. However, due to the specific nature of mathematical documents, mathematical topic classification is a challenging problem.

1.2.3 Human Expert Finding

A mathematical QA community should also support human expert finding for unanswered posted questions. It helps identify a list of experts who are most likely able to answer the posted questions based on their expertise. Human expert finding is an important function as unanswered questions could be delivered directly to the corresponding experts for answering which will enhance the rate of solved questions in the mathematical QA community. As such, it promotes users' satisfaction. Currently, there are many techniques available for expert finding including link analysis approach [34, 35, 36, 37] and content-based approach [3, 38]. Both existing approaches are promising. However, due to the specific nature of mathematical documents, the investigation of an expert finding approach for retrieving expert users for mathematics is quite challenging.

1.3 Objectives

Data mining [39, 40, 41, 42, 43], also known as Knowledge Discovery in Databases (KDD), aims to discover knowledge from large databases. Data mining techniques such as clustering, classification, association rule mining and sequential pattern mining have been applied to many areas including medical data analysis, credit card fraud detection, customer purchasing behavior prediction, Web access analysis and manufacturing process optimization [44].

Based on data mining techniques, the primary objective of this research is to investigate different techniques for supporting the functionalities of a mathematical QA community. To achieve this, this research will investigate the following issues as illustrated in Figure 1.2:

- *Mathematical Formula Retrieval*: Mathematical formula retrieval aims to find related formulas for an input formula query. In this research, we will investigate an approach for mathematical formula retrieval. In particular, we will focus on investigating an inverted indexing technique and a clustering-based technique for mathematical formula retrieval.

- *Mathematical Document Retrieval*: Mathematical document retrieval aims to find related mathematical documents for an input question query. In this research, we will investigate an effective mathematical document retrieval approach for an input question query according to its contents and formulas. In particular, we will focus on investigating a combined retrieval approach for both mathematical formula and text retrievals.

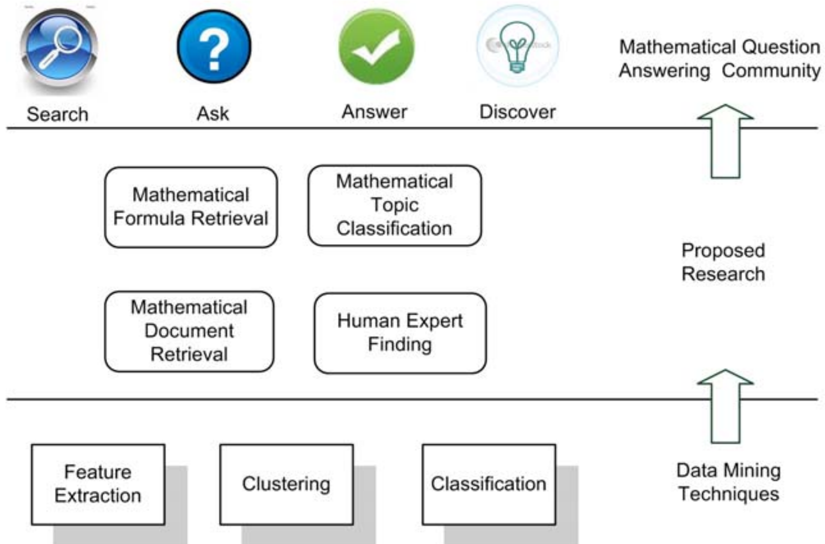


Figure 1.2: The Proposed Research Framework.

- *Mathematical Topic Classification*: Mathematical topic classification aims to help users to place their posted questions correctly into the corresponding topic categories. In this research, we will investigate an approach for mathematical topic classification for an input question query. In particular, we will focus on investigating a Support Vector Machine (SVM)-based technique [45] for mathematical topic classification.

- *Human Expert Finding*: Human expert finding aims to recommend a list of experts who may help solve users' posted questions according to their expertise. In this research, we will investigate an approach for human expert finding for an input question query. In particular, we will focus on investigating an index-based technique and a Latent Semantic Indexing (LSI)-based technique [46] for expert finding and retrieval.

As a result of this research, we have developed the Mathematical Question Answering (MathQA) Community with all the proposed techniques for mathematical formula retrieval, mathematical document retrieval, mathematical topic classification and human expert finding.

1.4 Organization of This Thesis

This chapter has introduced the motivation of this research. The objectives of the research have also been stated. The rest of the thesis is organized as follows. Chapter 2 reviews the related work on mathematical markup languages, Question Answering communities, mathematical search, document topic classification and human expert finding. Chapter 3 presents the proposed approach for formula retrieval and its performance evaluation. Chapter 4 presents the proposed approach for mathematical document retrieval and its performance evaluation. Chapter 5 presents the proposed approach for mathematical question topic classification and its performance evaluation. Chapter 6 presents the proposed approach for human expert finding and its performance evaluation. Finally, Chapter 7 concludes the thesis and discusses the directions for future research work.

Chapter 2

Related Work

In this chapter, we review the related work on mathematical markup languages, Question Answering communities, question search, document topic classification and human expert finding.

2.1 Mathematical Markup Languages

Mathematical formulas express the mathematical meanings through the use of mathematical symbols and structures. Different from textual descriptions, mathematical formulas cannot be simply encoded as an array of characters due to its mathematical contents such as functions, operators, variables and constant numbers. Figure 2.1 shows a mathematical formula containing mathematical symbols and operators. As can be seen, mathematical formulas are considered as two-dimensional information objects [47].

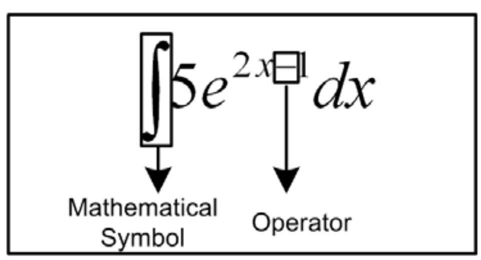


Figure 2.1: Mathematical Symbol and Operator in a Formula.

For representing formulas, a common approach is to use a standardized representation format such as *LaTeX* [48], *AsciiMath* [49] and *MathML* (Mathematical Markup Language) [50]. *LaTeX* is an extensively used markup language in the scientific community for document authoring. Formulas are represented as a sequence of text characters in this format. For example, the formula given in Figure 2.1 is represented as `$\int5e^{2x-1} dx$`. The *LaTeX* format is easy to use to represent formulas. Similar to *LaTeX*, *AsciiMath* also formats formulas as text characters and produces nice-looking formulas on Web pages. The formula shown in Figure 2.1 is formatted as ‘int5e^(2x-1) dx‘ in *AsciiMath*, which is generally simpler than that of *LaTeX*. Formulas represented in either *AsciiMath* or *LaTeX* are easy to store.

MathML is a standard markup language released as a recommendation by *W3C*. It is an *XML* application for describing mathematical notations. It captures both the semantics and structures in mathematical formulas. Using *MathML*, mathematical expressions are not only able to be represented semantically, but also can be displayed visually on Web browsers. The layout schemata of *MathML* are grouped into several classes. One group of elements is concerned with scripts which contain elements such as *msub*, *munder* and *mmultiscripts*. Another group focuses on more general layout which includes *mrow*, *mstyle* and *mfrac*. The third group deals with tables. Table 2.1 shows an example formula $\int 5e^{2x-1}dx$ encoded in *MathML*. Note that the symbol “$\int$“ in Table 2.1 is represented in its original form. But in practice, the symbol “$\int$“ is encoded using its *Unicode* representation. The *mo*, *mn*, *mi* and *msup* tags stand for the operators and $-$, the number constants 5, 2 and 1, the identifiers x and dx, and the *power* function respectively. In addition, the tag *mrow* is used to control the displaying format.

Table 2.1: The MathML Representation of the Formula $\int 5e^{2x-1}dx$.

MathML	**Meaning**
<math>	root
<mo> ∫ </mo>	∫ function
<mn> 5 </mn>	constant 5
<msup>	power function
<mi> e </mi>	identifier e
<mrow>	formatting tag
<mn> 2 </mn>	constant 2
<mi> x </mi>	identifier x
<mo> - </mo>	operator -
<mn> 1 </mn>	constant 1 (mn)
</mrow>	
</msup>	
<mrow>	formatting tag
<mi> dx </mi>	identifier dx (mi)
</mrow>	
</math>	

2.2 Question Answering Communities

There are many Question Answering (QA) communities available on the Internet. These QA communities can be grouped into two categories as follows:

- *Open Domain QA Communities*: It deals with question problems from many general topics such as health, science and education. There are many open domain QA communities available including Yahoo! Answers [9], Baidu Zhidao [10], TianYaWenDa [51], Google WenDa [52], FunAdvice [53], WikiAnswers [54], Sogou WenDa [55], Askville [56] and Answerbag [57]. Among these QA communities, Yahoo! Answers and Baidu Zhidao are two of the most popular open domain QA communities currently having millions of users.

- *Closed Domain QA Communities*: It deals with question problems for specific domains such as computer programming and telecommunications. There are many specialized closed domain QA communities available. These include The Math Forum at Drexel University [11] for the mathematics domain, Cramster [12] for most educational subjects such as mathematics, computer science and biology, LinkedIn Answers [58] for the business domain, DaTing [59] for selling and buying, Ask Bbioo [60] for the biology domain, TeLQAS [61] for telecommunication literature and CSDN [62] for the computer programming domain.

In this section, we will review some of the popular and related QA communities including Yahoo! Answers, Baidu Zhidao, The Math Forum at Drexel University and Cramster.

Yahoo! Answers [9] is a well-known and widely used open domain QA community which supports many categories such as sports, health, business, finance and arts. It provides an operational framework based on search, ask, answer and discover. All the posted questions in Yahoo! Answers are represented as textual data and organized into categories or topics. Yahoo! Answers provides text search to find related posted questions with answers. The text search can also be used for finding related mathematical problems. In this case, formulas can be entered in *AsciiMath* [49] like format for searching. Automatic question topic classification is provided to suggest the possible topics for the posted questions. To help find experts to solve unanswered questions, Yahoo! Answers uses the *reputation score* which is based on certain computations on the number of answers posted by each user. As such, the more reputation score a user has for a topic, the more likely he will be chosen as an expert for that topic.

Baidu Zhidao [10] is one of the most popular open domain QA systems in China. Similar to Yahoo! Answers, it supports many topics such as sports, health, business, finance and arts. It also provides the typical operational framework based on search, ask, answer and discover. In Baidu Zhidao, all the posted questions are treated as textual data. For question search, text retrieval technique is used for finding similar questions from the QA database, even for queries based on mathematical formulas. Similar to Yahoo! Answers, formulas can be entered in *AsciiMath* like format for searching. To help users categorize their posted questions, Baidu Zhidao provides a function on automatic question topic classification which suggests possible topics for a posted question. To help users find experts for unanswered questions, the *Baidu League* is created. Cuerrently, there are more than 260,000 members in the *Baidu League* [10]. The members are ranked according to their expertise in each topical domain. As such, experts can easily be identified based on the topics of the unanswered questions.

The Math Forum at Drexel University [11] is a comprehensive closed domain QA community for mathematics. Similar to the open domain QA communities, The Math Forum at Drexel University organizes all the questions based on mathematical topics and subtopics. The mathematical questions are processed as textual data and the mathematical formulas are also encoded using the *AsciiMath* like format. To search for similar questions, it supports the text search, formula search and integrated mathematical document search. In addition, a ranking score is also given in each retrieved question to indicate the relevance of the question to the input query. The Math Forum also provides many useful mathematics-related features and tools such as Math Help, Problems & Puzzles, Math Talk, etc. to enrich the functionalities of the mathematical QA community. However, automatic question topic classification is not supported. There is no automatic support to find experts for unanswered questions. However, The Math Forum provides a function called *Ask Dr. Math* which enables users to ask questions to the human experts directly. In fact, the experts are the staff who work in the School of Education of Drexel University.

Cramster [12] is a commercial QA community for many educational subjects such as mathematics, computer science, chemistry, physics, biology, etc. In particular, it aims to provide online homework help for college and high school students on most subjects including mathematics. Similar to other open domain QA communities, Cramster treats all input questions as textual data. For question search, only text retrieval is supported. As all input mathematical formulas are processed as images for browsing purposes, formula search is not supported. In addition, Cramster also does not provide the support for automatic topic classification of posted questions. To help users find experts to solve their problems, Cramster provides online support for users to contact human experts who are registered with the website. As Cramster is specifically designed for homework help on many educational subjects, it provides other useful features such as textbook help, study groups, lecture notes and online practice.

2.3 Question Search

In a QA community, question search aims to retrieve a set of related questions according to a user's input query. For most of the existing QA communities, question search is achieved by matching the keywords from the query and the stored questions using text retrieval techniques. Mathematical question search involves both question text search and mathematical search. In this section, we review the related work on question text search as well as mathematical search.

2.3.1 Question Text Search

In the past few decades, much work had been done on question text search. Burke et al. [63] proposed a FAQ FINDER system which uses both the statistical and semantic similarity measures to retrieve FAQs (Frequently-Asked Questions). To achieve this, the semantic knowledge base WordNet is used. In addition, FAQ FINDER [64, 65] also studied the impact of sense tagging and question typing for the retrieval of similar questions. In [66], Berger et al. proposed five statistical approaches to bridge the lexical chasm between questions and answers for answer-finding. Jijkoun et al. [67] proposed a vector space model based on Lucene [68] for QA (question-answering)-pair retrieval. To extract QA pairs from the fetched FAQ Web pages, heuristic extraction and machine learning extraction techniques are applied. In [69], Riezler et al. proposed two query expansion techniques based on the SMT (Statistical Machine Translation) technology for question search. A large corpus of QA pairs extracted from FAQ Web pages is used for learning a translation model from questions to answers. In [70], Soricut et al. applied MSNSearch and Google to search questions. To extract answers, the n-gram co-occurrence statistics and statistical translation techniques are used.

Duan et al. [71] proposed to use question topic information in a language modeling framework for question search. To automatically identify different question topics, the Minimum

Description Length (MDL) [72, 73] generalization method is used. Similarly, Cao et al. [74] proposed a framework that is capable of exploiting question categories for improving question search. Three millions QA pairs collected from Yahoo! Answers are used to test the proposed framework. Bian et al. [75] presented a ranking framework to retrieve relevant questions with the consideration of user interaction and feedback information. Moreover, data mining techniques such as clustering have also been explored for question text retrieval. In clustering-based retrieval, questions are converted into word vectors and grouped before query retrieval. Cao et al. [74] discussed the clustering-based retrieval approach which has two steps. The first step retrieves and returns one or more clusters of questions to a query [76]. In the second step, ranking scores of individual questions in the cluster are computed against the query [77, 78, 79, 80].

2.3.2 Mathematical Search

Currently, there are a number of mathematical search approaches such as Wolfram Formula Search [16], Wikipedia Formula Search [15], MathFind [14], Mathematical Document Retrieval [81], Whelp [13], ActiveMath [82], HELM [83], MBase [84], MathWeb [85], DLMF (Digital Library of Mathematical Functions) [86], MathDi [87], etc. Different from the text search approaches discussed above, most of mathematical search approaches focus on addressing the formula search problem. For some of these formula search approaches such as MathFind and Wikipedia Formula Search, similarly, they also employ the traditional text retrieval techniques. There are other approaches such as Wolfram Formula Search and Mathematical Document Retrieval which are based on semantic features extracted from formulas. In this section, we will review the following approaches: MathFind, Wikipedia Formula Search, Wolfram Formula Search and Mathematical Document Retrieval.

MathFind [14] is a mathematical document search system developed by Design Science [88]. In MathFind, all mathematical formulas are stored in the *MathML* format. For searching mathematical formulas, the search engine uses a formula processing layer implemented on top of a typical text-based search engine. The formula processing layer analyzes and decomposes the *MathML* formatted mathematical expressions into a sequence of text terms, which are called *math fragments*. These math fragments are analogous to the terms in text documents. In this manner, the common text terms as well as the math fragments are equally treated, and then indexed using the inverted indexing technique. For implementing the MathFind, the Apache Lucene Search [68] is used as the text-based search engine. To perform mathematical document search, MathFind combines the terms extracted from both formulas and text documents together. In MathFind, a graphical equation editor is provided for users to enter mathematical formula inputs for querying. The equation editor converts the input mathematical formula into its corresponding *MathML* format. A text box is also provided for entering text queries.

Wikipedia Formula Search [15] is another mathematical formula search engine. This formula search engine is specifically designed for searching mathematical formulas within the websites of Wikipedia [89], Wikipedia Japan and PlanetMath [90]. In Wikipedia Formula Search, all mathematical formulas are encoded using the *LaTeX* format. For parsing the mathematical formulas, it uses four different predefined word lists [15]. Based on the word lists, the formula search engine extracts meaningful mathematical elements such as $\backslash sin$ and $\backslash ln$ and filters out the formatted tags such as $\backslash begin$ and $\backslash end$ in *LaTeX*. After formula parsing, all mathematical formulas will be represented by text tokens which are then indexed using the inverted indexing technique. The indexing process is quite similar to that of the MathFind search engine. For formula retrieval, Wikipedia Formula Search provides a text box for users to enter their queries which are formatted in *LaTeX*. As a result, a list of related Web pages that contain the relevant formulas of the query are returned. In addition, each retrieved formula is assigned with a *percentage* to indicate its relevance to the input query.

Wolfram Formula Search [16] developed by Wolfram Research allows users to search a large database of mathematical formulas. A total of 122,544 mathematical formulas have been indexed which can be searched. All formulas in the Wolfram formula collection are stored in several different types of mathematical formats such as *Mathematica*-based [91] format and *MathML*. In particular, the Wolfram formula search engine focuses on indexing mathematical elements such as constant numbers and function terms of the mathematical formulas. Thus, most semantic features in the formulas can be extracted accordingly. Based on the different types of mathematical functions, e.g., *elementary functions* and *bessel-type functions*, the search engine can automatically organize all indexed formulas as a hierarchical tree structure. Each level of the tree is assigned with a predefined category label such as elementary functions, constants, bessel-type functions, numbers, etc. As such, users can specify a formula query by choosing the query's formula types from a drop-down menu. As a result, a list of retrieved formulas is returned and ranked based on certain criteria such as complexity.

In [81], Samarasinghe propose an approach for Mathematical Document Retrieval which is based on clustering techniques. The proposed approach treats formulas and text contents in mathematical documents separately for retrieval. Similar to that of MathFind, the proposed retrieval engine also uses the *MathML* format for encoding the mathematical formulas. For mathematical formula search, it first extracts the semantic features from the formulas based on a list of predefined *regular expressions*. Based on the extracted formula features, the formula search approach is then performed with two processes: semantic clustering and formula matching. In the semantic clustering process, the TFIDF [92] weighed formula semantic feature vectors are clustered using different clustering algorithms to generate the cluster models. During the querying process, the nearest cluster between each cluster centroid and the formula query is measured and identified. The formula matching process then calculates the Tree Edit Distances between the Mathematical Expression Trees constructed from the input formula query and each of the formulas in the retrieved cluster based on the Zhang-Shasha algorithm [93]. The retrieved formulas are then ranked according to the Tree Edit Distance and the cosine similarity scores. For text retrieval, it adopts the classical clustering-based text retrieval approach.

2.4 Document Topic Classification

Document topic classification aims to automatically categorize a given document into the appropriate topics or classes using classification algorithms such as Naïve Bayes (NB) [94], k-Nearest Neighbor (k-NN) [95], Support Vector Machine (SVM) [96], etc. Document topic classification has been applied to many areas such as emails, blogs and news articles.

2.4.1 Email Classification

Electronic Mail (email) has become one of the most popular communication mechanisms. One aspect of email classification is for filtering junk emails. As such, the email classification task can be viewed as a binary classification process: normal emails and abnormal emails. In [97], Martin et al. proposed an approach for classifying normal and junk emails. The proposed approach particularly focuses on analyzing the *behavioral features* of the normal and junk emails in order for classification. The behavioral features to be considered include the presence of HTML, presence of embedded images, number of attachments, etc. In addition, some other features such as the number of emails sent, number of unique email recipients, number of unique sender addresses, etc. are also considered. Finally, based on the extracted features, the SVM and NB classification algorithms are applied to achieve the classification task. In [98], Provost proposed an approach for sorting or classifying junk emails. The proposed approach is based on the bag-of-words model [99]. The emails are then transformed to the term frequency (TF) [100] weighed feature vectors, which consist of two parts: message body and message headers. In addition, the email addresses, domain names and URLs are also tokenized and treated as constituent "words". Finally, email classification was performed by using the NB and RIPPER [101] algorithms.

2.4.2 Blog Classification

Web log or *Blog* is a Web site that contains online personal journal with reflections, comments and often hyperlinks provided by the writer [102]. Recently, an increasing number of people prefer writing blogs for sharing ideas with others. According to [103], blog classification can be grouped into three types: blog identification (to determine whether a Web document is a blog), mood classification and genre classification.

Blog identification aims to identify the blog pages from a collection of Web pages. Therefore, similar to the email classification task, blog identification can also be viewed as a binary classification: blog or non-blog. In [104], Nanno et al. proposed a system that automatically collects and monitors blog collections, and identifies blog pages based on a number of heuristics. In [105], Elgersma and Rijke proposed a number of human-selected features such as comments to represent blogs. Then, text classification algorithms are applied for blog identification based on the blog features.

Mood classification is referred to as classifying blogs based on the mood at the level of individual post or a collection of posts. In [106], Mihalcea and Liu pointed out that there are two polarities of moods which can be derived from the blog contents: happiness and sadness. Thus, most blogs can be categorized into the two classes. The unigram text features are then extracted from the blog contents and blog classification is conducted using the NB algorithm. In [107], Chesley et al. categorized the blog posts into three sentiment classes: objective, positive and negative. Similarly, Mishne [108] used a binary classification approach for categorizing blogs into more than a hundred predefined moods. Here, the SVM classification technique is employed on training a variety of content and non-content features.

Blog genre classification generally categories blogs into genres [103] such as news, commentary and journal. In [109], Qu et al. proposed an approach for automatic classification of blogs into four genres: personal diary, news, political and sports. For representing the blog features, they use the unigram terms of the blog contents. Then, these text feature terms are transformed into TFIDF document vectors and fed into the NB algorithm for classification.

2.4.3 News Classification

Currently, online news articles are provided by many dedicated news wires such as Reuters [110] and PR Newswires [111]. In [112], Chan et al. proposed a classification system called Categorizer [112] that performs automated online news classification. The Categorizer adopts the SVM classification algorithm to classify news articles that are represented as TFIDF text feature vectors into categories. These categories can be either a set of predefined categories or special categories defined by users themselves. In [113], Masand et al. proposed a Memory Based Reasoning (MBR) classification approach for classifying hundreds of news stories based on seven different categorizes including industry, market sector, product, subject, government agency and region. They used the single words and capital word pairs as features, and the feature extraction task was done by a matching engine called Seeker. Billsus et al. [114] proposed an approach for classifying news stories based on a hybrid user model. Different from other approaches, the proposed approach aims to classify news stories into various topics based on the interests of individual users. The experimental results have shown that the proposed approach can achieve a much better accuracy than other algorithms such as k-NN and NB for news classification.

2.5 Human Expert Finding

Human expert finding looks for experts according to an input question query or in a specified topic. Human expert finding can be achieved by the following approaches: social network analysis approach, content analysis approach and their combination.

2.5.1 Social Network Analysis Approach

The Social Network Analysis technique [115, 116] can be used to find human experts based on a specified topic. It mainly focuses on analyzing the relationship information using the link analysis techniques. Two of the most popular link analysis algorithms are HITS [117, 118] and PageRank [119, 120].

Jurczyk et al. [34, 35] proposed the author ranking algorithm. They used the HITS algorithm to estimate the authority and evaluate the expected quality of users' Question Answer portals [9]. Zhang et al. [36] proposed a PageRank-like algorithm called *ExpertiseRank* for ranking the candidate experts. They analyzed *Java Forum* [36], a large online help seeking community using social network analysis techniques. Based on the results, they categorize the users into 5 expertise levels instead of a complete ranked list. And the social network analysis are particularly useful on corpora of email communications [121, 122, 123]. In [124], McLean et al. used a graph structure to propagate expertise evidence between members of a project team. The current approaches mainly focus on analyzing the relationships among all users, while the content information is generally ignored.

2.5.2 Content Analysis Approach

The content analysis aims to find human experts based on the user question query. This approach is mainly based on analyzing the question contents using traditional information retrieval (IR) techniques.

In [3], Liu et al. proposed an approach that uses the content of questions and answers to build expert profiles. It then finds the experts by comparing the similarities between the question contents and expert profiles. Similarly, Balog et al. [38] proposed two language models to search for experts on the Web. They viewed the task of expert finding as that of information retrieval. In [125], Macdonald et al. proposed a voting approach together with data fusion techniques across a range of document weighting models to find experts based on TREC 2005 [126]. To assess the effectiveness of the proposed data fusion techniques, three different document weighting models are applied. Pavel et al. [127] proposed a sequential dependency approach for expert finding. This approach assumes that sequential dependence exists between a candidate expert and the query terms of a document. Nick et al. [128] proposed the P@NOPTIC Expert system which automatically identifies experts based on the documents published on an organization's intranet. The system can be queried like a standard Web search engine, but it returns a list of experts rather than documents. Campbell et al. [37] proposed the expert finding approach in an email network, in which they had examined two algorithms for determining expertise from emails. The first is a content-based approach that considers only the text contents, while the second uses a graph-based ranking algorithm (HITS) that considers both the text contents and communication patterns.

2.5.3 Combined Approach

The combined approach finds human experts using both network analysis and content analysis approaches.

Zhang et al. [129] proposed a propagation-based approach to identify the potential experts from an academic researcher network. The approach is divided into two steps: Initialization and Propagation. In the Initialization step, each participant's profile, contact information and publications are extracted as *local information* and formed a *document*. In the Propagation step, they use the person relationship information to improve the accuracy of expert finding based on the propagation theory [130]. Finally, the scores for the experts rely on the scores earned from the above two steps. Zhou et al. [131] proposed novel approaches to find experts in online forums, which consists of three steps: Model Generation, Index Creation and Re-ranking. Model Generation generates three different types of models for computing the user expertise, namely profile-based model, thread-based model and cluster-based model. Index Creation builds inverted indexes for above generated models. This step aims to enhance the performance brought by calculating the ranking values with three models, especially since multiple users may pose questions simultaneously. The final step is to compute the re-ranking score using the structural relations between users in a forum, in which a PageRank algorithm is employed. Chen et al. [132] proposed five types of relations between two users that may influence the reputation when building a social network graph, in which each type of edge has a different weight. Then, a reputation computation mechanism is proposed to compute the influence when the graph is changed. Zhang et al. [133] proposed an approach that not only compares the similarity of questions with user profiles, but also considers other detailed factors such as query posting time and replies. Kao et al. [134] proposed a more comprehensive approach for finding experts based on Yahoo! Answers in Taiwan [135]. They considered almost all possible factors for ranking answerers including the contents of question and answers pair, the voting value, the evaluation factor, the time decay, the user's reputation and the people's relationship.

2.6 Summary

In this chapter, we have reviewed the related work on mathematical markup languages, question search, document topic classification and human expert finding. The related techniques are useful for consideration for the development of the proposed Mathematical Question Answering (MathQA) Community. For mathematical question search, we will investigate both the inverted indexing technique and clustering-based technique for mathematical formula and document retrieval. The inverted indexing technique is efficient and easy to maintain, whereas clustering-based technique has been successfully applied to many document clustering tasks. For mathematical topic classification, we will investigate the different classification techniques which are traditionally used for the classification task. In particular, we will investigate the Support Vector Machine which is able to achieve better performance when compared with other classification techniques for many automatic classification applications. For human expert finding, the content-based approach is quite suitable for our application. Therefore, we will investigate the content-based approach for expert finding in the MathQA Community.

Chapter 3

Mathematical Formula Retrieval

Mathematical formulas are highly symbolic and structured. Mathematical formula retrieval aims to help users to search similar formulas in the MathQA Community. It retrieves related formulas based on mathematical formula features from an input formula query. In this chapter, we propose an approach for mathematical formula retrieval. The proposed approach and its performance evaluation will be presented.

3.1 Mathematical Formula Retrieval in MathQA Community

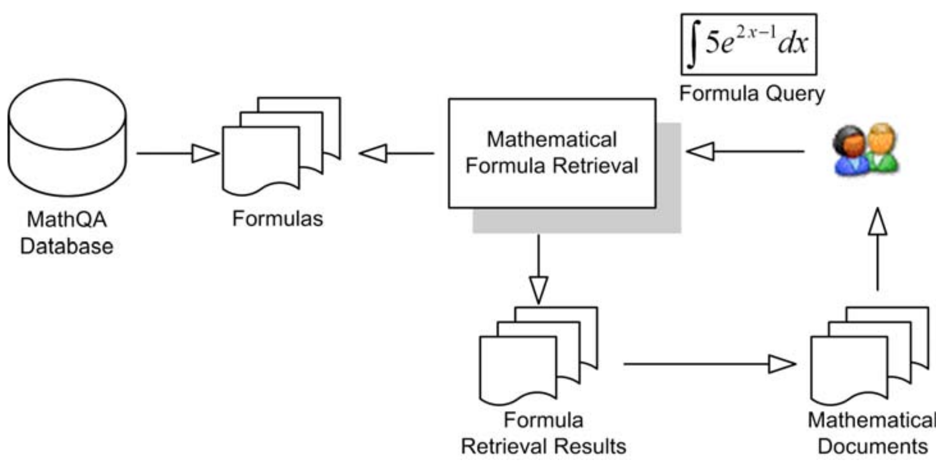


Figure 3.1: Mathematical Formula Retrieval in MathQA Community.

Figure 3.1 shows the mathematical formula retrieval process in the MathQA Community. First, an input mathematical formula is entered by a user as a query. Then, based on the proposed formula retrieval approach, it retrieves a set of related formulas from the MathQA Database according to the mathematical features between the stored formulas and the formula query. Finally, the mathematical documents containing the related formulas can be retrieved and displayed.

3.2 Proposed Approach

Figure 3.2 shows the proposed approach for mathematical formula retrieval. It consists of the following processes:

- *Formula Feature Identification*: It identifies useful features from mathematical formulas.
- *Formula Feature Extraction*: It extracts the useful features from mathematical formulas.
- *Index-based Retrieval*: It performs formula retrieval using the inverted indexing technique based on the extracted formula features.
- *Clustering-based Retrieval*: It performs formula retrieval using the classical clustering algorithms based on the extracted formula features.

In the following sections, we will discuss each process in details.

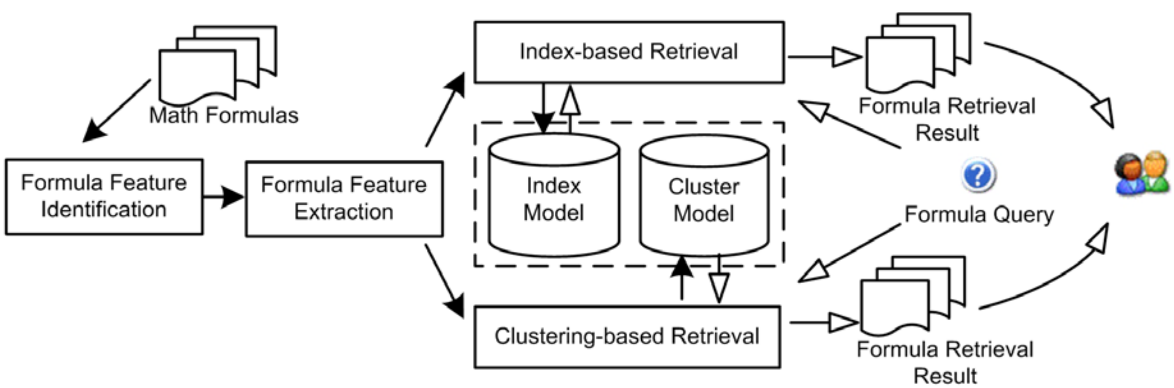


Figure 3.2: Proposed Formula Retrieval Approach.

3.3 Formula Feature Identification

Figure 3.3 shows an example formula on $\int 5e^{2x-1}dx$. As shown in the example formula, it contains four types of formula features which are described as follows:

- *Semantic Feature*: It refers to the semantic information in a formula including its *functions* and *operators*. For example, the semantic features in the formula $\int 5e^{2x-1}dx$

include the integration function ($\int dx$), the exponential function (e), the power function and the subtraction operator (-).

- *Structural Feature*: It refers to the structural information among the elements in a formula. For example, the structural feature in the formula $\int 5e^{2x-1}dx$ includes the structural relationship between e and $(2x-1)$ with a *power* function.

- *Constant Feature*: It refers to the constant information in a formula. For example, the constant features in the formula $\int 5e^{2x-1}dx$ are 5, 2 and 1.

- *Variable Feature*: It refers to the variable information in a formula. For example, the variable feature in the formula $\int 5e^{2x-1}dx$ is x.

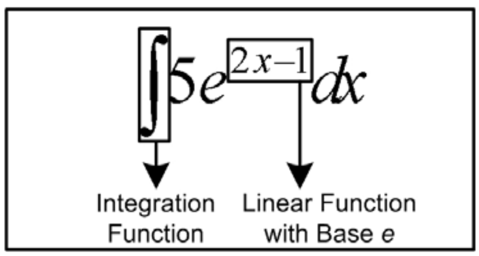


Figure 3.3: Formula Features of $\int 5e^{2x-1}dx$.

3.4 Formula Feature Extraction

The formula feature extraction process is used to extract the four types of formula features, namely semantic features, structural features, constant features and variable features. Figure 3.4 shows the Formula Feature Extraction process, which consists of five steps: AsciiMath to MathML Conversion, DOM Tree Construction, Semantic Feature Extraction, Structural Feature Extraction, and Constant and Variable Feature Extraction.

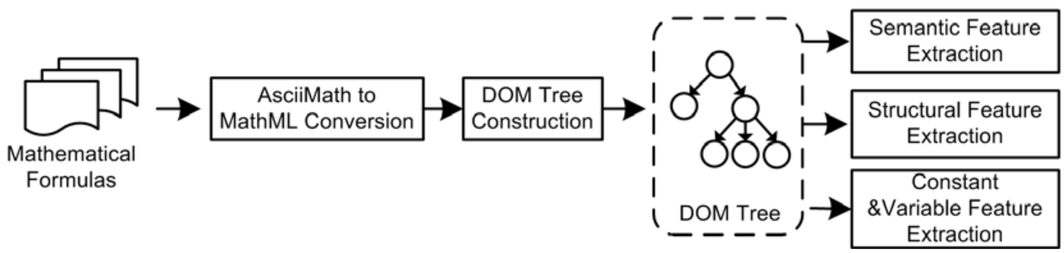


Figure 3.4: Formula Feature Extraction.

3.4.1 AsciiMath to MathML Conversion

As the *AsciiMath* formula markup representation is quite convenient to store and manage, we store all formulas in the *AsciiMath* format. However, the major drawback of the *AsciiMath* representation is that it is not easy to process and extract the four types of formula features as all formula elements are encoded as a sequence of characters. On the other hand, the *MathML* formula markup representation is based on the *XML* format, which can be parsed and processed quite easily. Table 3.1 shows the *MathML* and *AsciiMath* markups for the example formula $\int 5e^{2x-1}dx$. Therefore, in this step, we convert all formulas into *MathML* representation from the *AsciiMath* format for subsequent processing. To achieve this, *AsciiMathPHP* [136] is used for the conversion.

Table 3.1: AsciiMath and MathML Markups for $\int 5e^{2x-1}dx$.

Formula	AsciiMath	MathML
$\int 5e^{2x-1}dx$	int5e^(2x-1)dx	<math>
		<mo> ∫ </mo>
		<mn> 5 </mn>
		<msup>
		<mi> e </mi>
		<mrow>
		<mn> 2 </mn>
		<mi> x </mi>
		<mo> - </mo>
		<mn> 1 </mn>
		</mrow>
		</msup>
		<mrow>
		<mi> dx </mi>
		</mrow>
		</math>

3.4.2 DOM Tree Construction

In this step, we first construct the *Document Object Model* (DOM) tree structure for a given formula encoded by *MathML*. Then, all the subsequent feature extraction steps will be carried out based on the DOM tree. Algorithm 3.1 describes the DOM tree construction process.

To illustrate the operations of Algorithm 3.1, we take the *MathML* of the formula $\int 5e^{2x-1}dx$ given in Table 3.1 as an example. As shown in Figure 3.5, firstly, we create the *root* of the DOM tree by using the first start-tag $< math >$ and set this node as CP (in step 1 of Figure

Algorithm 3.1 DOM_Tree_Construction

Input:

MathML - A formula in *MathML* format

Output:

DOM Tree - A tree structure representing a formula

Process:

1: Create the *root* of the DOM Tree for the first $MathML$ tag and set it as $Current\ Parent$ (CP).
2: Starting from the next tag, check whether it is the start-tag or end-tag. If it is the start-tag, create a *tree node* by the name of this $MathML$ tag, add it as a *child* under CP and set this newly created *tree node* as CP. Otherwise, add the tag's content as *inner text* of CP and set CP's *parent* as CP.
3: Recursively create and add nodes to the DOM Tree following steps 1 and 2 until the end-tag of the *root* is reached.
4: **return** DOM Tree.

3.5). Then, as the next tag $< mo >$ is a start-tag, we create a *tree node* named *mo*, add it as a child to the CP ($< math >$), and set this newly created node as CP. As the following tag is an end-tag, its content ($\int$) is then added as the *inner text* to the CP mo as $mo[\int]$, and its *parent* ($math$) is set as CP. This process continues until it reaches the end-tag of the *root* (i.e., $< /math >$).

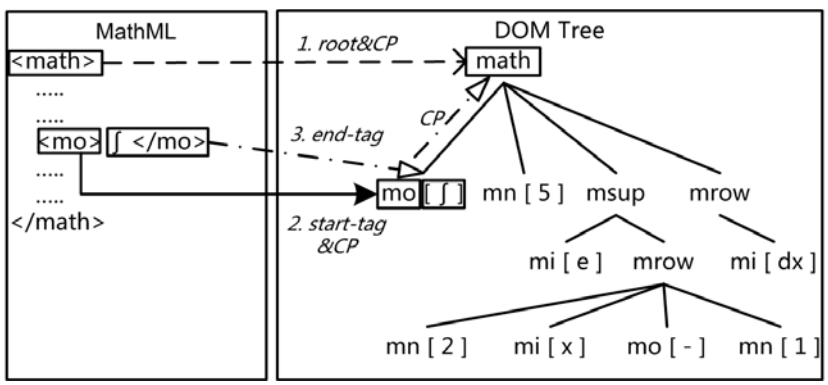


Figure 3.5: An Example on DOM Tree Construction.

3.4.3 Semantic Feature Extraction

This step aims to extract the semantic features from a given DOM tree. Note that the semantic features extracted in this step are referring to the formula functions and operators. Given a DOM tree, the semantic feature extraction process is performed by traversing the entire tree in preorder and extracting the corresponding contents, (e.g., *name* or *inner text*) of each *tree node* as semantic features. Algorithm 3.2 presents the algorithm for semantic feature extraction.

Algorithm 3.2 Semantic_Feature_Extraction

Input:

DOM Tree - A tree structure representing a formula

Output:

{Semantic_features} - A set of semantic features of a formula

Process:

1: Traverse individual *tree nodes* in DOM Tree in preorder except the *root*.
2: If the *name* of the *tree node* is *mo*, add its *inner text* to {Semantic_ features}.
3: Else if the *name* of the *tree node* equals to *mi*, although it means *identifier*, we still have to check whether there is any formula functions in its *inner text* or not. If it has, add the node's *inner text* to {Semantic_features}.
4: Else check whether there is any *tree node* named *mi* among all its *children*. If there is, add the node's *name* to {Semantic_features} except for the formatting node (*mrow*).
5: **return** {Semantic_features}.

For example, to extract the semantic features from the DOM tree shown in Figure 3.5, we begin with the *tree node* $mo[\int]$. As it satisfies the condition in step 2 of Algorithm 3.2, the symbol $\int$ is extracted as a semantic feature representing the *integration function* (see Figure 3.6(a)). Likewise, for the *tree node* $mi[\mathrm{e}]$, after checking its *inner text*, as the symbol e represents the *exponential function*, we extract it as a semantic feature. However, in order to extract the semantic feature for the *tree node* *msup*, we find one of its *children* $mi[x]$ containing a variable x according to step 4 of Algorithm 3.2. Hence, the *msup* is retained as a semantic feature (see Figure 3.6(b)). Finally, the resultant semantic features for the DOM tree given in Figure 3.5 are extracted as $\{\int, msup, e, -\}$.

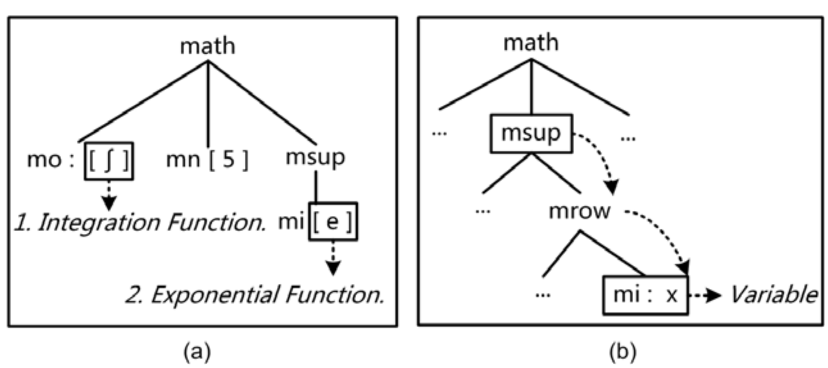


Figure 3.6: An Example on Semantic Feature Extraction.

3.4.4 Structural Feature Extraction

This step aims to extract the formula structural features from a given DOM tree. Formula structural features are also important in determining the characteristics of a formula. Consider two formulas $f^2(x)$ and $ff(x)$, although both formulas have the same symbol f, its meaning is rather different in each formula. In $f^2(x)$, the f has a *square*, which means f has the power of 2. While in $ff(x)$, the two fs are concatenated together. Thus, the semantic meaning of the symbol f relies on not only its semantic feature as *function*, but also its structural features (e.g., f has the power of 2). Algorithm 3.3 presents the algorithm for structural feature extraction.

Algorithm 3.3 Structural_Feature_Extraction

Input:

DOM Tree - A tree structure representing a formula

Output:

{Structural_features} - A set of structural features of a formula

Process:

1: Traverse each *tree node* in the DOM Tree in preorder except the *root*.
2: If the node contains semantic feature and its *depth* is greater than one (*depth* equals one means no relation to others), traverse its *parent*, *grandparent*, etc. until reaching the *root*. Then, combine the *names* of these traversed nodes with their semantic features together and add these combined terms into {Structural_features}.
3: **return** {Structural_features}.

For example, consider the structural feature extraction process for the *tree node* $mi[-]$ ($depth = 3 > 1$) of the DOM tree given in Figure 3.5. Firstly, using Algorithm 3.3, we extract its semantic feature as $-$. Secondly, we visit its *parent* (*mrow*) and *grandparent* (*msup*) until the *root* (*math*) is reached. Finally, we combine *mrow* and *msup* with the semantic features to form the structural feature as $msup\$mrow\$-$ ("$" is a separator). Figure 3.7 illustrates the above process. As a result, the resultant structural features for the DOM tree given in Figure 3.5 are extracted as $\{msup\$e, msup\$mrow\$-\}$.

3.4.5 Constant and Variable Feature Extraction

As both formula constant and variable are often structurally related to other elements, for example, in the formula $(x+1)^2$, the number constant 2 is related to the sub-formula $(x+1)$ with the structural meaning of "$(x+1)$ to the power of 2". Thus, it is better to describe such information with structural features. Therefore, we use Algorithm 3.3 to extract the features

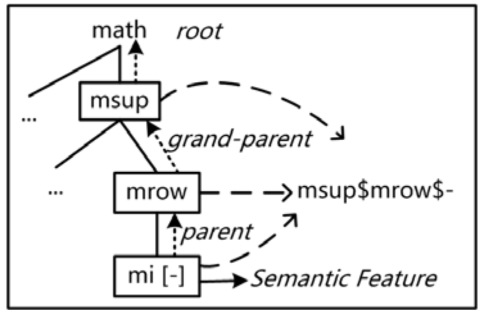


Figure 3.7: An Example on Structural Feature Extraction.

for both constants and variables. However, there are two differences from Structural Feature Extraction. Firstly, the extracted values of the variables and constants are represented as common terms as *var* and *cn* (stand for variable and constant respectively) rather than their exact values. Secondly, the constant and variable features are still extracted when the *depth* of the *tree node* equals to one.

For example, the constant and variable feature extraction process for the DOM tree given in Figure 3.5 can be described as follows. Firstly, we extract the structural features as $msup\$mrow\$2$, $msup\$mrow\$1$, and $msup\$mrow\$x$ by Algorithm 3.3. Then, we replace the exact values with the terms *var* and *cn* accordingly. As the depth of the *tree node* $mi[5]$ equals to 1, the constant feature for the constant 5 is then extracted as *cn*. As a result, the constant and variable features for the DOM tree given in Figure 3.5 are extracted as $\{cn$, $msup\$mrow\$cn$, $msup\$mrow\$var\}$.

3.5 Index-based Retrieval

This section presents the proposed index-based technique for formula retrieval. The index-based technique which is shown in Figure 3.8 consists of the following two processes: Indexing and Query Retrieval.

The Indexing process aims to create the index model using the inverted indexing technique. It consists of the following three steps:

- *Semantic Index Terms Generation*;
- *Other Index Terms Generation*; and
- *Index Model Generation*.

The Query Retrieval process aims to retrieve a set of related and ranked formulas to an input formula query based on the created index model. It comprises the following three steps:

- *Query Feature Extraction*;
- *Related Formula Retrieval*; and
- *Formula Ranking*.

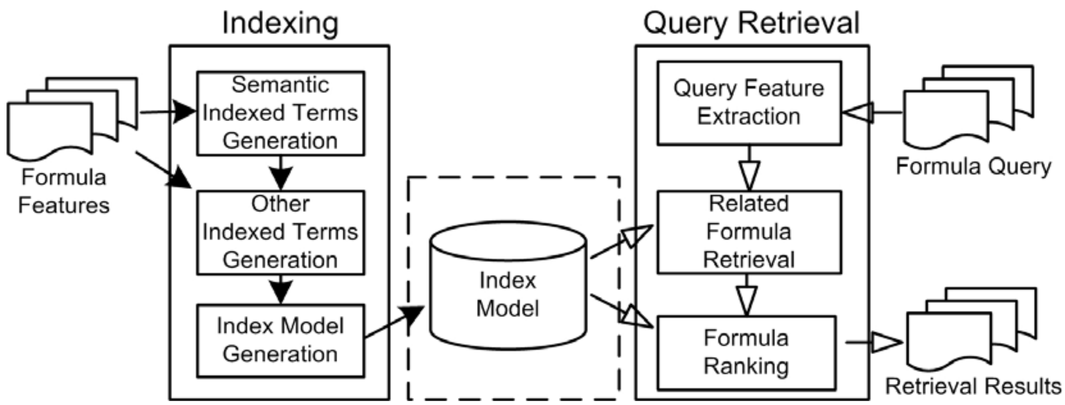


Figure 3.8: Proposed Index-based Retrieval Technique.

3.5.1 Semantic Index Terms Generation

In this research, we generate two types of index terms for formula semantic features: *in-order semantic terms* and *sorted semantic terms*.

In-Order Semantic Terms Generation

In-order semantic terms are generated using n-grams from the original ordered sequence of formula semantic features. As the in-order semantic terms retain the order information of a formula, this is useful for ranking the retrieved formulas according to the query using the order information (this will be explained later in Section 3.5.5). To generate the in-order semantic terms for a formula, we apply the n-gram technique to all the formula's extracted semantic features, where $n = 2$, 3 and 4.

For example, the extracted semantic features for the formula $\int 5e^{2x-1}dx$ are $\{\int$, $msup$, e, and $-\}$. Using the n-gram technique, the generated in-order semantic terms are obtained as follows:

- 2-gram = $\{ \int \$msup, msup\$e, e\$- \}$;

- 3-gram = { $\int$ \$$msup$\$$e$, $msup$\$$e$\$$-$ }; and
- 4-gram = { $\int$ \$$msup$\$$e$\$$-$ } .

where "\$" is a separator. As a result, the six in-order semantic terms for the formula $\int 5e^{2x-1}dx$ are: {$\int$ \$$msup$, $msup$\$$e$, e\$$-$, $\int$ \$$msup$\$$e$, $msup$\$$e$\$$-$, $\int$ \$$msup$\$$e$\$$-$}. Note that the 1-gram in-order semantic terms do not contain order information, e.g., $msup$ and e. Hence, n starts from 2.

Sorted Semantic Terms Generation

Sorted semantic terms are generated based on the lexicographically sorted semantic feature sequence. As the sorted semantic terms have removed the order information after sorting, they are useful for retrieving related formulas (this will be explained later in Section 3.5.5). To generate the sorted semantic terms, the only difference from that of the In-Order Semantic Terms Generation is that we first sort the semantic features in their lexicographical order before applying the n-gram technique, where n starts from 1 to 4.

For example, after sorting the semantic features of the formula $\int 5e^{2x-1}dx$, the sorted terms obtained are {e, $msup$, $-$, $\int$ }. For those functional terms, we use their corresponding *Unicode* representations for comparison. Afterwards, using the n-gram technique, the four sets of sorted semantic terms are obtained as follows:

- 1-gram = { e, $msup$, $-$, $\int$ };
- 2-gram = { e\$$msup$, $msup$\$$-$, $-$\$$\int$ };
- 3-gram = { e\$$msup$\$$-$, $msup$\$ $-$ \$$\int$ }; and
- 4-gram = { $msup$\$$e$\$ $-$ \$$\int$ }.

As a result, ten sorted semantic terms for the formula $\int 5e^{2x-1}dx$ are generated as: {e, $msup$, $-$, $\int$, e\$$msup$, $msup$\$$-$, $-$\$$\int$, e\$$msup$\$$-$, $msup$\$ $-$ \$$\int$, $msup$\$$e$\$ $-$ \$$\int$ }.

3.5.2 Other Index Terms Generation

For other formula features such as structural features, constant features and variable features, their feature terms are treated as index terms directly. Table 3.2 summarizes all the generated index terms from the formula $\int 5e^{2x-1}dx$.

Table 3.2: Example Index Terms of the Formula $\int 5e^{2x-1}dx$.

Formula	Index Terms	Description
$\int 5e^{2x-1}dx$	∫$msup, msup$e, e$-, ∫$msup$e, msup$e$-, ∫$msup$e$-	In-Order Semantic Terms
	e, musp, -, ∫, e$msup, msup$-, -$∫, e$msup$-, msup$-$∫, msup$e$-$∫	Sorted Semantic Terms
	msup$e, msup$mrow$-	Structural Terms
	cn, msup$mrow$cn	Constant Terms
	msup$mrow$var	Variable Terms

3.5.3 Index Model Generation

In this step, all the generated index terms including semantic index terms and other index terms are stored as inverted indexes [92].

3.5.4 Query Feature Extraction

Given a query formula, this step extracts its formula features using the proposed formula feature extraction algorithms discussed in Section 3.4. As a result, the query's semantic features, structural features, constant features and variable features are obtained in preparation for the Related Formula Retrieval step.

3.5.5 Related Formula Retrieval

As semantic features contain the most useful information of a formula, we use the semantic features to retrieve the related formulas according to an input query. As mentioned, two types of index terms are generated from semantic features, namely in-order semantic terms and sorted semantic terms. Here, we use the sorted semantic terms for formula retrieval since they can achieve better retrieval performance compared with that of the in-order semantic terms.

For example, consider the following three formulas: $\cos x \sin x \ln x$, $\ln x \cos x \sin x$ and $\sin x \cos x \ln x$, and the query $\sin x \cos x \ln x$.

- *Using Sorted Semantic Terms*: As the three formulas share exactly the same semantic features: *cos*, *ln* and *sin*, after sorting, all of them will share the same sorted semantic terms, i.e., *sin*, *cos*, *ln* , *cos*$*sin*, *cos*$*ln* and *cos*$*sin*$*ln*. During the retrieval process, if we use the query's 3-gram sorted semantic term *cos*$*sin*$*ln*, all the three formulas are retrieved as all have *cos*$*sin*$*ln*. Thus, the retrieval process only requires one matching operation.

- *Using In-Order Semantic Terms*: As this type of terms retain the order information, the three formulas have different sets of 2-gram and 3-gram in-order semantic terms as follows:

  - $\cos x \sin x \ln x$: { *cos\$sin*, *sin\$ln*, *cos\$sin\$ln* };
  - $\ln x \cos x \sin x$: { *ln\$cos*, *cos\$sin*, *ln\$cos\$sin* }; and
  - $\sin x \cos x \ln x$: { *sin\$cos*, *cos\$ln*, *sin\$cos\$ln* }.

  If we use the query's 3-gram in-order semantic terms (*sin\$cos\$ln*) to carry out the matching operation, only the last formula will be retrieved. The other two will be retrieved unless the query's 1-gram semantic terms (i.e., *sin*, *cos* and *ln*) are used. In this case, it requires 3 matching operations.

The index-based retrieval technique is basically based on inverted indexes. However, the *intersection* operation, also known as the *AND* operation, used in the traditional text document retrieval approach may not be appropriate for formula retrieval simply because of the unique characteristics of formulas. For example, consider the following three formulas: $\sin x$, $\sin x + 1$ and $\sin x - 2$. Intuitively, these formulas share a common function term *sin*. If $\sin x + 1$ is used as a query, the other two formulas should be considered as relevant. But if we use *intersection*, nothing would be matched as the other formulas do not contain the operator $+$. For this reason, we have used the *union* operation on sorted semantic terms to retrieve the query's related formulas. However, one major concern of the *union* operation is that the retrieval efficiency may be affected due to the drastic increase in the number of matched items. To deal with this problem, we use the *top k retrieval* [92] technique as used in Google, where k is initially set to 10 by default. In this way, the results could be returned as soon as the number of candidates reaches k. As such, the negative effect caused by the use of the *union* operation is minimized.

Algorithm 3.4 illustrates the Related Formula Retrieval process. The overall process is quite straightforward. In the third line, we generate n-gram semantic terms by decreasing the order of n, where n is from N (maximum) to 1 (minimum). This is to ensure that formulas which share more semantic features with the query can be retrieved first. In line 11 and line 12, we return the result as soon as the number of retrieved formulas reaches k.

Algorithm 3.4 Related_Formula_Retrieval

Input:

{Semantic_features} - A set of semantic features of a query

N - The maximum number n used in the n-gram technique

k - The number of formulas to be retrieved

Output:

{Related_formulas} - A set of related formulas

Process:

Initialize {Related_formulas} $\leftarrow \emptyset$
{Semantic_features} $\leftarrow$ *Sort_by_lexicographic_order* ({Semantic_features})
for $i = 1$ **to** N **do**
  {Sorted_semantic_terms} $\leftarrow$ *Generate_n-gram*({Semantic_features}, i)
  /*generate i sorted semantic terms.*/
  for all *term* $\in$ {Sorted_semantic_terms} **do**
    {formulas} $\leftarrow$ *Retrieve_formulas_from_inverted_indexes*(*term*)
    {Related_formulas} $\leftarrow$ {Related_formulas}$\cup${formulas}
    /* *union* retrieved formulas.*/
  end for
  if $|${Related_formulas}$| \geq k$ **then**
    return {Related_formulas}
    /*k related formulas have been obtained.*/
  end if
end for
return {Related_formulas}

3.5.6 Formula Ranking

In order to rank the retrieved formulas, we compute a *matching score* based on the different types of matched index terms with the query. Then, all the retrieved formulas will be ranked according to their matching scores.

Semantic Feature Matching Score

Semantic features have two types of index terms, i.e., in-order semantic terms and sorted semantic terms. As the in-order semantic terms retain the order information of a formula, we use the in-order semantic terms to judge the relevancy between each retrieved formula and the query in terms of the order information. While for judging the semantic relevancy between each retrieved formula and the query, we use the 1-gram sorted semantic terms.

According to the mathematical nature and importance of the semantic information of the 1-gram sorted semantic terms, they can be generally classified into two types: operator terms and function terms. The operator terms are mathematical operators, while the function terms are mathematical functions. Compared with the latter, the operator terms are less important in determining formula ranking. Table 3.3 shows some examples of both types of terms.

Table 3.3: Examples on Operator and Function Terms.

Operator Terms	**Meaning**	**Function Terms**	**Meaning**
$+$	addition	sin	sine function
$-$	subtraction	cos	cosine function
$\times$	multiplication	sum	summation
$mfrac$	division	log	logarithm
$=$	equality	$msqrt$	square root

In order to determine the weight carried by each term, we have used the Inverse Document Frequency (IDF) [137] weighing scheme. Let N be the total number of formulas in a collection. The IDF value for feature term t is given as follows:

$$\mathrm{IDF}(t) = \log \frac{N}{\mathrm{DF}(t)}, \quad \mathrm{DF} \neq 0 \tag{3.1}$$

where DF is the Document (formula) Frequency. Note that the denominator DF (t) denotes the number of formulas in the collection that contain the feature term t. Thus, the IDF for rare terms are higher than those that occur frequently.

Based on the IDF weighing scheme, the semantic feature matching score for each retrieved formula can be computed through the following four steps:

1. If any retrieved formulas contain a query's operator terms, we add the IDF value of the operator terms to its semantic feature matching score. In this case, the semantic feature matching score for formula f to query q is obtained as follows:

$$\sum_{t_j \in S_{OT}(f,q)} \mathrm{IDF}(t_j) \tag{3.2}$$

where S_{OT} is the set of matched operator terms.

2. If any retrieved formulas contain a query's function terms, we add not only their IDF values, but also the maximum IDF value of the collection, i.e., $logN$. This is obtained by setting the denominator of DF to its minimum value (i.e., 1) in Equation (3.1). In this case, the semantic feature matching score for formula f to query q is obtained as follows:

$$(\sum_{t_j \in S_{FT}(f,q)} \mathrm{IDF}(t_j) + |S_{FT}(f,q)| \times \log N \tag{3.3}$$

where S_{FT} is the set of matched function terms and N is the total number of formulas in the collection.

3. If any retrieved formulas do not contain a query's operator terms or function terms, or contain additional terms, we take away the IDF value of the unmatched terms from the semantic feature matching score. In this case, the semantic feature matching score for formula f to query q is obtained as follows:

$$-\sum_{t_j \in S_{UM}(f,q)} \mathrm{IDF}(t_j) \tag{3.4}$$

where S_{UM} is the set of unmatched terms.

4. If any retrieved formulas contain a query's in-order semantic terms, we add the IDF value of the matched in-order semantic terms to its semantic feature matching score. In this case, the semantic feature matching score for formula f to query q is obtained as follows:

$$\sum_{t_j \in S_{IT}(f,q)} \mathrm{IDF}(t_j) \tag{3.5}$$

where S_{IT} is the set of matched in-order semantic terms.

In summary, the total semantic feature matching score for formula f to query q is given as follows:

$$\begin{aligned} Score_{semantic}(f,q) = & \sum_{t_j \in S_{OT}(f,q)} \mathrm{IDF}(t_j) + \sum_{t_j \in S_{FT}(f,q)} \mathrm{IDF}(t_j) \\ & + |S_{FT}(f,q)| \times \log N - \sum_{t_j \in S_{UM}(f,q)} \mathrm{IDF}(t_j) + \sum_{t_j \in S_{IT}(f,q)} \mathrm{IDF}(t_j) \end{aligned} \tag{3.6}$$

where S_{OT} is the set of matched operator terms, S_{FT} is the set of matched function terms, S_{UM} is the set of unmatched terms, S_{IT} is the set of matched semantic in-order terms and N is the total number of formulas in the collection.

Structural Feature Matching Score

To compute the structural feature matching score for a retrieved formula, we only add the IDF value of the matched structural terms. In fact, it is quite similar to the step 4 in computing semantic feature matching score. Thus, the structural feature matching score for formula f to query q is given as follows:

$$Score_{structural}(f,q) = \sum_{t_j \in S_{SF}(f,q)} \mathrm{IDF}(t_j) \tag{3.7}$$

where S_{SF} is the set of matched structural terms.

Constant and Variable Feature Matching Scores

To compute both the constant and variable feature matching scores, we use the number of matched constant and variable terms rather than their IDF values. Thus, the constant and variable feature matching scores for formula f to query q are obtained as follows:

$$Score_{constant}(f,q) = |S_{MT}(f,q)| \tag{3.8}$$

$$Score_{variable}(f,q) = |S_{MT}(f,q)| \tag{3.9}$$

where S_{MT} is the set of matched constant or variable terms.

Matching Score Normalization

From the above computation, we can see that the more the query formula features a retrieved formula has, the higher the formula matching score is. Therefore, the next step is to normalize the matching score. The normalized semantic feature matching score $||Score_{semantic}(f,q)||_{norm}$ is given as follows:

$$||Score_{semantic}(f,q)||_{norm} = \frac{Score_{semantic}(f,q)}{Score_{semantic}(q,q)} \tag{3.10}$$

where $Score_{semantic}(q,q)$ is the total semantic feature matching score obtained by matching the query formula q to itself. It is computed as follows:

$$
\begin{aligned}
Score_{semantic}(q,q) = &|S_{OT}(q,q)| \times \mathrm{IDF}(t_j) + |S_{FT}(q,q)| \times \mathrm{IDF}(t_j) \\
&+ |S_{FT}(q,q)| \times \log N + |S_{IT}(q,q)| \times \mathrm{IDF}(t_j)
\end{aligned}
$$

where $|S_{OT}(q,q)|$ is the number of query's operator terms, $|S_{FT}(q,q)|$ is the number of query's function terms, $|S_{IT}(q,q)|$ is the number of query's semantic in-order terms, and N is the total number of formulas in the collection. Note that the query will be matched exactly to itself. Thus, the step 3 in computing semantic feature matching score is 0.

Finally, the normalized matching score for formula f to query q for each type of feature is given as follows:

$$
||Score_n(f,q)||_{norm} = \frac{Score_n(f,q)}{Score_n(q,q)} \tag{3.11}
$$

where n is semantic, structural, constant or variable.

As both constant and variable features are less important in determining the ranking compared with that of semantic features and structural features, we add a parameter α to discriminate their contributions to the total matching score. The total matching score for formula f to query q is given as follows:

$$
\begin{aligned}
Score_{formula}(f,q) = (1-\alpha) \times &\frac{||Score_{semantic}(f,q)||_{norm} + ||Score_{structural}(f,q)||_{norm}}{2} + \\
\alpha \times &\frac{||Score_{constant}(f,q)||_{norm} + ||Score_{variable}(f,q)||_{norm}}{2} \\
&(\text{where } 0 \leq \alpha \leq 1)
\end{aligned} \tag{3.12}
$$

In this research, the parameter α is set to 0.2 which is determined experimentally (to be discussed in Section 3.7.3).

An Example

Figure 3.9 shows an example query and two retrieved formulas. Table 3.4 gives the IDF values of the 1-gram sorted semantic terms and the matched index terms. And Table 3.5 shows the matched index terms. The total number of formulas in a collection is 884, i.e., $N = 884$.

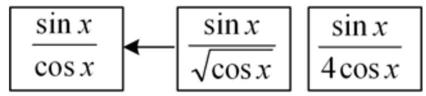


Figure 3.9: An Example Query and Two Retrieved Formulas.

Table 3.4: IDF Values for 1-gram Sorted Semantic Terms and Matched Index Terms.

Terms	IDF	Description
sin	1.17	1-gram Sorted Semantic Terms
cos	1.10	
msqrt	1.84	
mfrac	0.80	
sin$mfrac	1.71	Matched In-Order Semantic Terms
mfrac$cos	1.82	
mfrac$mrow$sin	1.86	Matched Structural Terms
mfrac$mrow$cos	1.10	

Table 3.5: Matched Index Terms.

Formula	Matched In-Order Semantic Terms	Matched Structural Terms	Matched Constant or Variable Terms
$\frac{\sin x}{\sqrt{\cos x}}$	sin$mfrac	mfrac$mrow$sin	mfrac$mrow$var
$\frac{\sin x}{4\cos x}$	sin$mfrac mfrac$cos sin$mfrac$cos	mfrac$mrow$sin	mfrac$mrow$var

Step 1: Computing the semantic feature matching score:

$Score_{semantic}(\frac{\sin x}{\sqrt{\cos x}}, \frac{\sin x}{\cos x}) = \mathrm{IDF}(mfrac) + [\mathrm{IDF}(sin) + \mathrm{IDF}(cos) + 2log(884)] - \mathrm{IDF}(msqrt) + \mathrm{IDF}(sin\$mfrac)$

$$= 0.80 + (1.17 + 1.10 + 5.89) - 1.84 + 1.71 = 8.83$$

$Score_{semantic}(\frac{\sin x}{4\cos x}, \frac{\sin x}{\cos x}) = \mathrm{IDF}(mfrac) + [\mathrm{IDF}(sin) + \mathrm{IDF}(cos) + 2log(884)] - 0 + (\mathrm{IDF}(sin\$mfrac) + \mathrm{IDF}(mfrac\$cos) + \mathrm{IDF}(sin\$mfrac\$cos))$

$$= 0.80 + (1.17 + 1.10 + 5.89) - 0 + (1.71 + 1.82 + 1.93) = 14.42$$

$Score_{semantic}(\frac{\sin x}{\cos x}, \frac{\sin x}{\cos x}) = \mathrm{IDF}(mfrac) + [\mathrm{IDF}(sin) + \mathrm{IDF}(cos) + 2log(884)] + (\mathrm{IDF}(sin\$mfrac) + \mathrm{IDF}(mfrac\$cos) + \mathrm{IDF}(sin\$mfrac\$cos))$

$$= 0.80 + (1.17 + 1.10 + 5.89) + (1.71 + 1.82 + 1.93) = 14.42$$

$$||Score_{semantic}(\tfrac{\sin x}{\sqrt{\cos x}}, \tfrac{\sin x}{\cos x})||_{norm} = \tfrac{8.83}{14.42} = 0.61$$

$$||Score_{semantic}(\tfrac{\sin x}{4\cos x}, \tfrac{\sin x}{\cos x})||_{norm} = \tfrac{14.42}{14.42} = 1.00$$

Step 2: Computing the structural feature matching score:

$$Score_{structural}(\tfrac{\sin x}{\sqrt{\cos x}}, \tfrac{\sin x}{\cos x}) = \mathrm{IDF}(mfrac\$mrow\$sin) = 1.86$$

$$Score_{structural}(\tfrac{\sin x}{4\cos x}, \tfrac{\sin x}{\cos x}) = \mathrm{IDF}(mfrac\$mrow\$sin) = 1.86$$

$$Score_{structural}(\tfrac{\sin x}{\cos x}, \tfrac{\sin x}{\cos x}) = \mathrm{IDF}(mfrac\$mrow\$cos) + \mathrm{IDF}(mfrac\$mrow\$sin)$$

$$= 1.86 + 1.10 = 2.96$$

$$||Score_{structural}(\tfrac{\sin x}{\sqrt{\cos x}}, \tfrac{\sin x}{\cos x})||_{norm} = \tfrac{1.86}{2.96} = 0.63$$

$$||Score_{structural}(\tfrac{\sin x}{4\cos x}, \tfrac{\sin x}{\cos x})||_{norm} = \tfrac{1.86}{2.96} = 0.63$$

Step 3: Computing the constant feature matching score:.

As there is no constant in the query, the constant feature matching scores for the retrieved formulas are 0.

Step 4: Computing the variable feature matching score:

$$Score_{variable}(\tfrac{\sin x}{\sqrt{\cos x}}, \tfrac{\sin x}{\cos x}) = 1.00$$

$$Score_{variable}(\tfrac{\sin x}{4\cos x}, \tfrac{\sin x}{\cos x}) = 1.00$$

$$Score_{variable}(\tfrac{\sin x}{\cos x}, \tfrac{\sin x}{\cos x}) = 1.00 + 1.00 = 2.00$$

$$||Score_{variable}(\tfrac{\sin x}{\sqrt{\cos x}}, \tfrac{\sin x}{\cos x})||_{norm} = \tfrac{1.00}{2.00} = 0.50$$

$$||Score_{variable}(\tfrac{\sin x}{4\cos x}, \tfrac{\sin x}{\cos x})||_{norm} = \tfrac{1.00}{2.00} = 0.50$$

Step 5: Computing the total matching score:

$$Score_{formula}(\tfrac{\sin x}{\sqrt{\cos x}}, \tfrac{\sin x}{\cos x}) = (1.00 - 0.20) \times \tfrac{0.61+0.63}{2} + 0.20 \times \tfrac{0.00+0.50}{2} = \mathbf{0.55}$$

$$Score_{formula}(\tfrac{\sin x}{4\cos x}, \tfrac{\sin x}{\cos x}) = (1.00 - 0.20) \times \tfrac{1.00+0.63}{2} + 0.20 \times \tfrac{0.00+1.00}{2} = \mathbf{0.75}$$

Retrieval Results:

The ranked order for the two retrieved formulas is $\frac{\sin x}{4\cos x}$ and $\frac{\sin x}{\sqrt{\cos x}}$.

3.6 Clustering-based Retrieval

In this section, we present a clustering-based technique for formula retrieval. In this research, formula clustering is conducted using three different types of clustering algorithms, namely K-means [138], Agglomerative Hierarchical Clustering (AHC) [139] and Kohonen's Self-Organizing Map (SOM) [140]. These algorithms are chosen as they are typical clustering algorithms which are commonly used for document clustering and retrieval [141, 142, 143, 76, 144, 145, 146, 147, 148].

Figure 3.10 shows the clustering-based retrieval technique which comprises the following two processes:

- *Training*: It trains cluster models using the three clustering algorithms. This process consists of two steps: Transformation and Cluster Model Generation.
- *Query Retrieval*: It retrieves a set of related formulas according to an input formula query based on the generated cluster models. This process comprises three steps: Query Feature Extraction, Query Transformation, and Cluster Selection and Ranking.

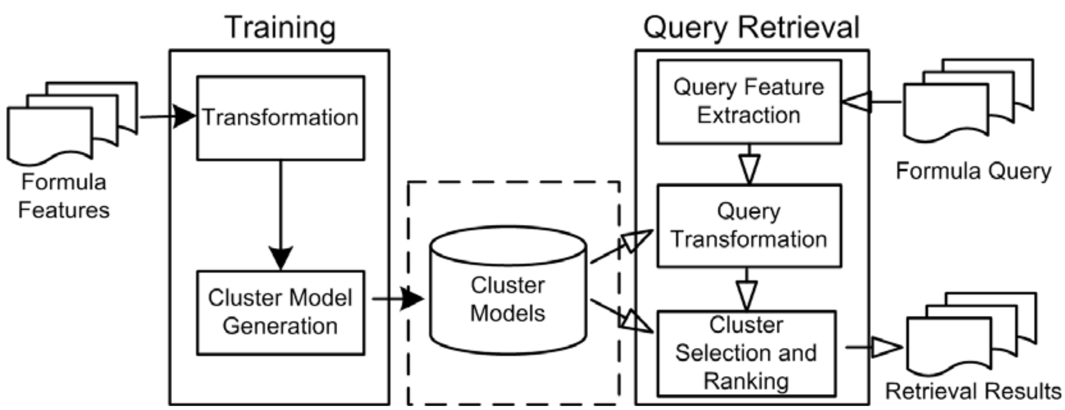


Figure 3.10: Clustering-based Retrieval.

3.6.1 Transformation

This step converts all the formula features obtained from the Formula Feature Extraction step into vector representations, which are then normalized in preparation for cluster model generation. To do this, we first identify a list of formula feature terms (1-gram) from a collection of formulas. The list contains a total of 1074 distinct formula feature terms, including 75 semantic terms, 478 structural terms, 304 constant terms and 217 variable terms. Each term is indexed with a sequence number, which indicates its position in the list. Based on the list, we then map the feature terms into a 1×1074 vector and the value in each dimension is weighed by TFIDF [92]. Thus, the input formulas are transformed into their corresponding vector representations. Finally, we normalize these TFIDF feature vectors before feeding

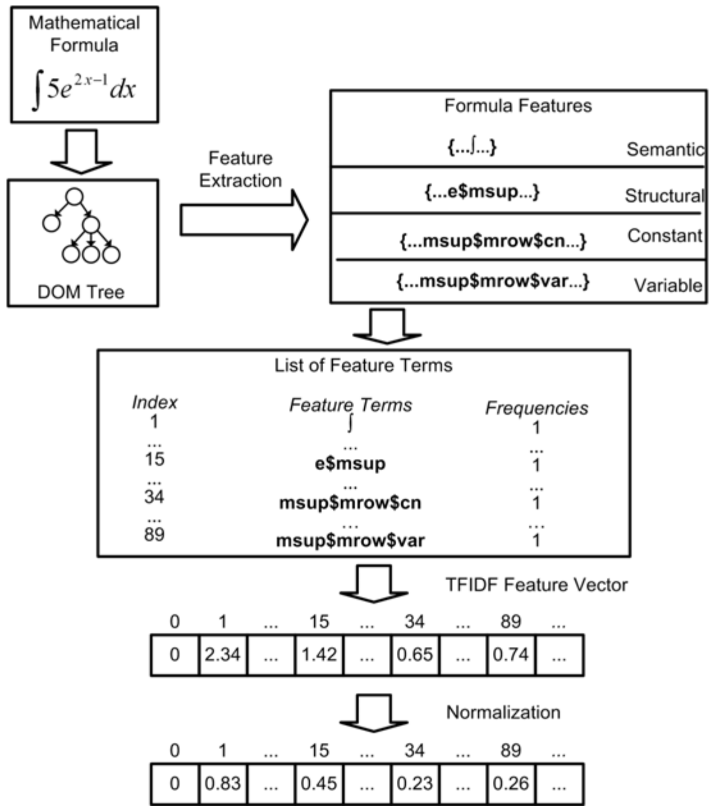


Figure 3.11: An Example on Transformation.

them into the clustering algorithms for training purpose. This is done by using *cosine normalization* [92] which converts the original vector $\vec{V}(w_1, w_2, ..., w_M)$ to its normalized vector $\vec{V}_{norm}$ as follows:

$$\vec{V}_{norm} = \frac{1}{\sqrt{w_1^2 + w_2^2 + ... + w_M^2}}\vec{V} = \frac{1}{\sqrt{w_1^2 + w_2^2 + ... + w_M^2}}(w_1, w_2, ..., w_M) \tag{3.13}$$

Figure 3.11 illustrates an example on the Transformation process.

3.6.2 Cluster Model Generation

In this step, the cluster model for each clustering algorithm is generated and stored for the subsequent Query Retrieval process.

In this research, we explore three different types of clustering algorithms including K-means, Agglomerative Hierarchical Clustering (AHC) and Kohonen's Self-Organizing Map (SOM). In order to generate the cluster models, we have to train the three clustering algorithms independently as each of them has a different training process.

K-means Algorithm

The training process for the K-means algorithm can be found in [149]. In order to generate the cluster model for the K-means algorithm, we have to determine the parameter K, i.e., the number of initial centroids. Various methods have been proposed in the literature for centroid initialization, and the simplest method is to randomly initialize the K centroids [150]. However, the performance of K-means is highly subjective to the selection of the initial K centroids [150]. That is, different runs of K for the same input may produce different results [151]. Therefore, in this research, the K-means algorithm is trained for different values of K, in the range from 5 to 100 in intervals of 5 (i.e., 5, 10, 15,.., 100).

Agglomerative Hierarchical Clustering Algorithm

The training process for the Agglomerative Hierarchical Clustering (AHC) algorithm can be found in [152]. For the AHC algorithm model generation, two parameters need to be initialized beforehand, i.e., the proximity calculation method and the stopping criterion. There are three such proximity calculation methods, namely single link, complete link and the group average method. As for the stopping criterion, either a threshold value or the desired number of clusters can be used. The group average proximity calculation method balances the pros and cons of both single link and complete link methods [92]. In addition, it has also been the most widely adopted method among the three proximity calculation methods [153]. Therefore, we have used the group average method to measure the proximity. In this research, we set the stopping criterion of the AHC algorithm based on the number of desired clusters from 5 to 100 in intervals of 5 (i.e., 5, 10, 15,.., 100).

Kohonen's Self-Organizing Map Neural Network

The training process for the Kohonen's Self-Organizing Map (SOM) algorithm is given in [140]. Compared with the model generation processes for the K-means and AHC algorithms, the training process for the SOM algorithm is much more complicated as there are totally seven parameters needed to be decided beforehand. The seven parameters for SOM include the dimensionality of the SOM grid, the shape of the SOM grid, the initial learning rate, the learning rate adjusting function, the neighborhood function, the neighborhood updating function and the number of training iterations. It is rather hard to obtain all the optimal parameters which can achieve the best retrieval performance and there are no fixed rules for setting any of these training parameters [140]. Generally, there are two ways to train the cluster models for SOM [154]. The first way is to set the dimensionality and fix the shapes of the SOM grid (i.e., the number of neurons is fixed). Then, the SOM algorithm is trained with different numbers of training iterations. In the second way, the number of training iterations is fixed and the SOM algorithm is trained for different numbers of neurons (i.e., different dimensionalities and shapes). In this research, based on [154], the numbers on iterations and neurons are set to 400 and 20 respectively.

3.6.3 Query Feature Extraction

It is similar to the same step in the index-based retrieval technique. This step extracts the four types of formula features for the Query Retrieval process using the formula feature extraction algorithms discussed in Section 3.4.

3.6.4 Query Transformation

Similar to the Transformation step in the training process, this step transforms the different formula features of the input query formula into a vector representation.

However, there is one difference for this Query Transformation step. Instead of weighing the feature vector by TFIDF as in the training process, the query vector is weighed using $qf \times$ IDF [155], where qf is the frequency of occurrence of a feature term in the query formula and IDF is the Inverse Document Frequency of the same feature term in the formula collection. The reason is that a term that occurs more frequently in the query is likely to be more important than those which occur infrequently, and the terms that occur infrequently in the formula collection are likely to be more important than the frequent terms.

3.6.5 Cluster Selection and Ranking

After Query Transformation, the generated normalized query vector is used to find clusters which contain the most similar mathematical formulas. To do this, the cluster centroid vectors stored in the Cluster Model database are retrieved, and the Euclidean distance $\mathrm{dist}(\vec{V}(q), \vec{V}(c))$ is computed between the normalized query vector $\vec{V}(q)$ and a cluster centroid vector $\vec{V}(c)$ using the following formula:

$$\mathrm{dist}(\vec{V}(q), \vec{V}(c)) = \sqrt{\sum_{i=1}^{n} (\vec{V}(q_i) - \vec{V}(c_i))^2} \tag{3.14}$$

where n is the number of dimensions in the normalized query vector, and q_i and c_i are the weights of the i^{th} element in the normalized vector and the centroid vector respectively. The smaller the Euclidean distance, the more likely the cluster will contain formulas similar to the formula query.

Once the nearest cluster is found, then the ranking step is carried out by computing the *cosine similarity* [92] between each formula vector in the cluster and the query vector by the following equation:

$$\mathrm{sim}(\vec{V}(q), \vec{V}(f)) = \frac{\vec{V}(q) \cdot \vec{f}(f)}{|\vec{V}(q)| \cdot |\vec{V}(f)|} \tag{3.15}$$

where the numerator represents the dot product (also known as the inner product) of the vectors $\vec{V}(q)$ and $\vec{V}(f)$, while the denominator is the product of their *Euclidean Lengths*. In fact, both formula and query vectors have been normalized. Thus, we only need to compute the Euclidean distance.

3.7 Performance Evaluation

This section presents the performance evaluation of the proposed formula retrieval approach.

3.7.1 Dataset

As there is no benchmark formula dataset available for conducting the performance evaluation, we have created the Gaokao Formula Dataset by collecting mathematical formulas from 88 past mathematics examination papers from the year of 2004 to 2008 of the National Higher Education Entrance Examination (or Gaokao) [156]. In this dataset, there are a total of 5501 mathematical formulas. Based on the formula types, it falls into 16 categorizes such as *inequality*, *trigonometry* and *series*. Table 3.6 summarizes these formula types and presents some sample mathematical formulas. Note that the sample formulas in Table 3.6 are presented using its original forms in the Gaokao examinations.

3.7.2 Evaluation Measures

In this section, the performance of the proposed formula retrieval approach is evaluated based on retrieval accuracy including Precision at 5 (P@5), Precision at 10 (P@10) and Mean Average Precision (MAP) [157]. They are defined as follows:

Table 3.6: Formula Types and Samples in Gaokao Formula Dataset.

Formula Types	Total	Sample Formulas
Exponential Function	41	$y = e^{\frac{1}{2}x}$
Trigonometry	328	$\sin A + \cos A = \frac{\sqrt{2}}{2}$
Limits	59	$\lim_{n\to\infty} x_n = 2$
Geometry Equation	421	$\alpha \perp \beta, \alpha \parallel \beta$
Series	259	$a_{n+1} = a_n + c_n$
Function	555	$f(x_1 + x_2) = f(x_1) * f(x_2)$
Vector	368	$vec(c) = vec(a) + vec(b)$
Complex Number	104	$z = \frac{1}{2+i}$
Summation	12	$\sum_{i=1}^{n} T_i < \frac{2}{3}$
Inequality	664	$\frac{1}{a} + \frac{1}{b} + \frac{1}{c} < \frac{7}{8}$
Absolute Value	82	$\|2 + x\| \geq \|x\|$
Logarithmic Function	110	$c = \log_2 \sin(\frac{2}{5}\pi)$
Proportion	127	$AE : EB = AF : FC = 2:1$
Set Theory	76	$(A \cup B) \cap C$
Point	437	$P(x_0, y_0)$
Others	1858	$A_1B_1C_1, \alpha, \beta$
Total	**5501**	

- *Precision at k* (P@k): P@k is the fraction of the top k retrieved items that are relevant to the query. It is defined as follows:

$$Precision = \frac{\#\text{relevant items}}{\#\text{retrieved items}} \tag{3.16}$$

- *Mean Average Precision* (MAP): MAP is the mean average precision for the top k retrieved items. It has the same meaning as another measurement called Average Precision at Seen Relevant Documents [158]. This metric favors retrieval systems that retrieve relevant documents with higher ranking. It is a good indicator for the usability of each retrieval approach, since users typically spend more time in examining top ranked results than those lower ones. MAP is given as follows:

$$MAP = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{m_i}\sum_{k=1}^{m_i} Precision(rank_{ik}) \tag{3.17}$$

where Q is the set of queries; $rank_{ik}$ is the set of ranked retrieved items starting from the top item until reaching item i_k; m_i is the number of items retrieved for query i; $Precision(rank_{ik})$ is the precision of query i at a given cut-off rank k, i.e., the ratio of the top k retrieved items that are relevant.

3.7.3 Experiments

To conduct the experiments, we divide the Gaokao Formula Dataset into training and test sets. By choosing 6 formulas from each formula type, we have 96 formulas for testing. The remaining 5405 formulas are then used for training. We further divide the testing data into two sets. The first half of the testing data which consists of 48 formulas is used as the development set to tune the parameters for both index-based and clustering-based retrieval techniques. This is done by choosing 3 formulas from each formula type. The second half which consists of another 48 formulas is used for performance evaluation with the tuned parameters. Table 3.7 summaries the number of formulas used for training and testing.

Table 3.7: Number of Formulas for Training and Testing.

	Training Data	**Testing Data**		**Total**
		Development Set	***Test Set***	
Number of Formulas	5405	48	48	5501

Parameter Tuning

In this section, we tune the parameters for both index-based and clustering-based retrieval techniques. For the index-based technique, the control parameter α needs to be tuned. On the other hand, the parameters used for the K-means and AHC algorithms in the clustering-based techniques also need to be tuned. We use the Mean Average Precision (MAP) for tuning the parameters, and then the parameters with the best MAP performance will be chosen for performance evaluation on the test set.

For the K-means clustering algorithm, the initial number of centroids (K) should be tuned. As mentioned in Section 3.6.2, we set K from 5 to 100 in intervals of 5. Figure 3.12 shows the performance results of the different values of K based on MAP for the K-means algorithm. From Figure 3.12, we can see that the MAP performance changes slightly with different values of K. The K-means algorithm achieves the best MAP performance of 82.13% when K is set to 15. In addition, we also observe that the MAP performance for K-means becomes stable at around 77.54% after K is set to 70 or higher. Thus, the best value for K in K-means is obtained at 15.

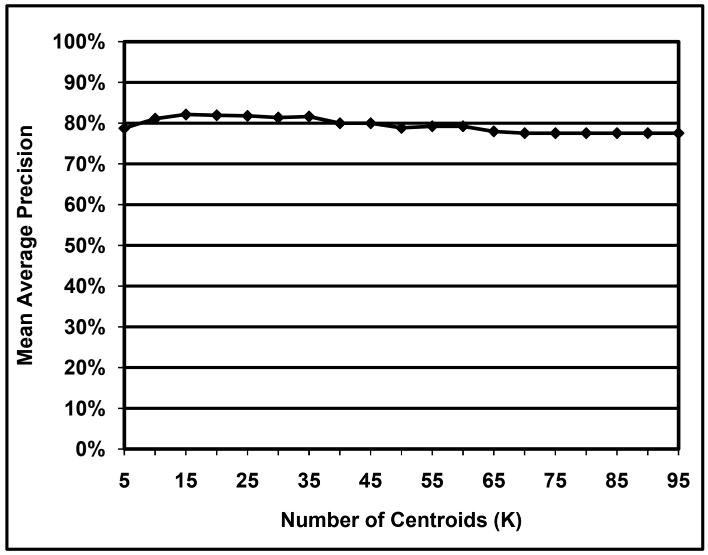


Figure 3.12: Performance Results based on Different Values of K in K-means.

For the AHC clustering algorithm, the stopping criterion (i.e., the number of desired clusters) has to be tuned. Similar to the parameter tuning in K-means, we set the stopping criterion from 5 to 100 in intervals of 5. Figure 3.13 shows the performance results of the different values of the stopping criterion based on MAP for the AHC algorithm. From Figure 3.13, we observe that the MAP performance increases slightly until reaching the best value at 76.55% when the stopping criterion is set to 20. Afterwards, the MAP performance becomes stable at around 72.37%. Therefore, the best value for the stopping criterion is obtained at 20.

For the index-based retrieval technique, the control parameter α needs to be tuned. The parameter α is used to discriminate the contributions of the semantic and structural features, and the constant and variable features to the total matching score given in Equation (3.12). In the experiment, we set the value of α from 0 to 1 in intervals of 0.1. Figure 3.14 shows the performance results based on MAP using different values of α. As can be seen from Figure 3.14, the MAP performance is greatly influenced by α. After reaching the best MAP performance at 88.04% with α setting to 0.2, the MAP performance decreases drastically until reaching its lowest MAP performance at 62.13%. Therefore, we choose $\alpha = 0.2$.

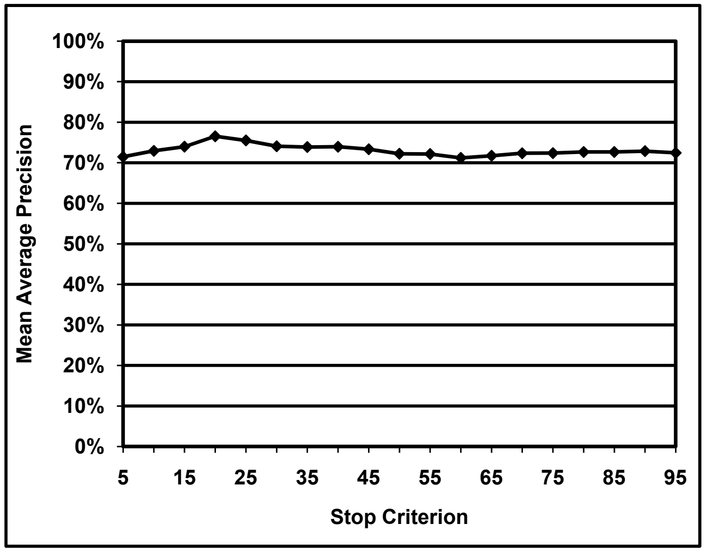


Figure 3.13: Performance Results based on Different Values of the Stopping Criterion in AHC.

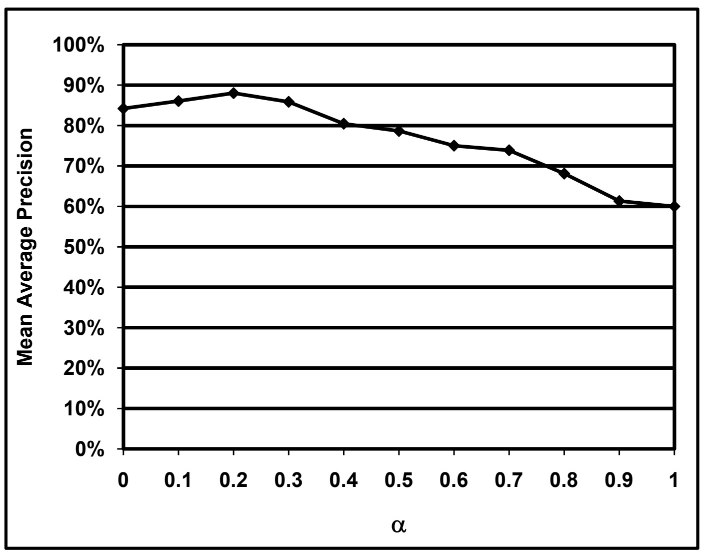


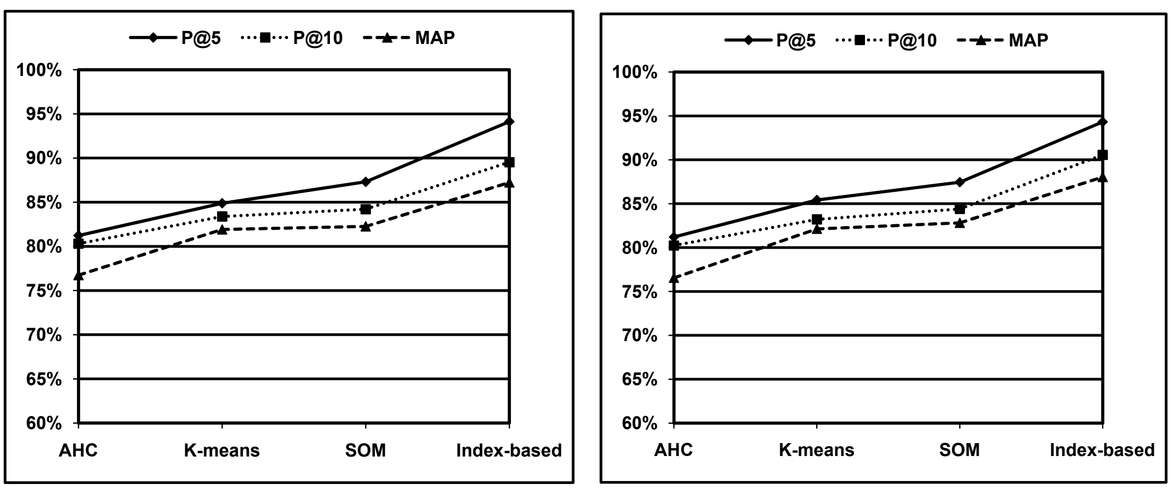
Figure 3.14: Performance Results based on Different Values of α in the Index-based Technique.

Performance Results

In this section, we evaluate the performance for both the clustering-based retrieval technique and the index-based retrieval technique using the formulas in the test set. For K-means and AHC, we use the best parameter values to conduct the performance evaluation. In addition, the performance evaluation based on the development set is also given. The performance results are given in Table 3.8 and Figure 3.15. From the performance results, we can see that the index-based retrieval technique has achieved better performance than the clustering-based technique for both development set and test set. For the test set, the best performance results of the index-based retrieval technique are 94.12%, 89.53% and 87.23% for P@5, P@10 and MAP respectively. Among the three clustering algorithms, SOM has achieved the best performance results for both the development set and test set.

Table 3.8: Formula Retrieval Performances on Development Set and Test Set.

	Development Set				Test Set		
	P@5	*P@10*	*MAP*		*P@5*	*P@10*	*MAP*
Index-based	94.32%	90.56%	88.04%		94.12%	89.53%	87.23%
K-means	85.42%	83.21%	82.13%		84.88%	83.38%	81.91%
AHC	81.20%	80.24%	76.55%		81.23%	80.31%	76.74%
SOM	87.45%	84.40%	82.81%		87.31%	84.21%	82.26%



(a) Development Set (b) Test Set

Figure 3.15: Formula Retrieval Performance on Development Set and Test Set.

3.8 Summary

In this chapter, we have presented our proposed approach for mathematical formula retrieval. In the proposed approach, formula feature identification and extraction are important in order to extract indicative formula features for the subsequent retrieval process. We have also developed two different techniques for formula retrieval based on the index-based technique and clustering-based technique. The performance of the proposed approach based on the two techniques is also evaluated and compared. The performance results have shown that the index-based technique has achieved better performance than the clustering-based retrieval technique.

Chapter 4

Mathematical Document Retrieval

Mathematical documents contain not only textual data, but also mathematical formulas. Mathematical document retrieval aims to help users to search for similar mathematical documents in the MathQA Community. It retrieves related mathematical documents based on both textual and formula features for an input mathematical document query. In this chapter, we propose an index-based mathematical document retrieval. The proposed approach and its performance evaluation will be presented.

4.1 Mathematical Document Retrieval in MathQA Community

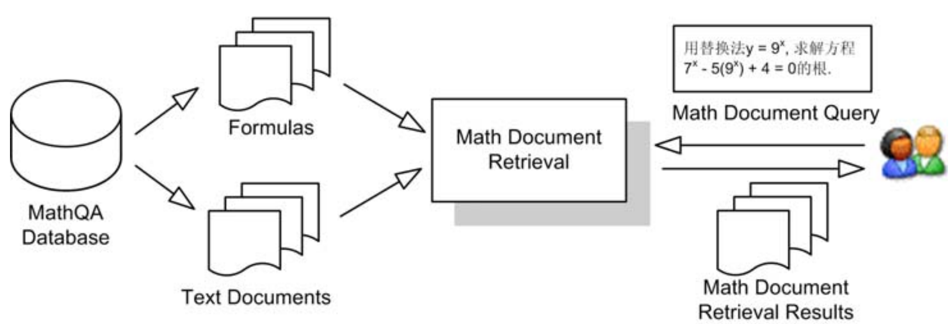


Figure 4.1: Mathematical Document Retrieval in MathQA Community.

Figure 4.1 shows the mathematical document retrieval process in the MathQA Community. First, an input mathematical document is entered by a user as a query. Note that the mathematical document query contains only the question part as the answer is not available during the time of querying. Then, based on the proposed index-based mathematical document retrieval approach, it retrieves a set of related mathematical documents from the MathQA Database according to both the textual and mathematical features between the stored documents and the document query. Finally, the related mathematical documents can be retrieved and displayed.

4.2 Proposed Approach

Figure 4.2 shows the proposed mathematical document retrieval approach. It consists of the following processes:

- *Formula and Text Separation*: It separates the text and formula components from the mathematical documents.
- *Formula Indexing and Retrieval*: It retrieves a set of related mathematical documents based on an input formula query. The index-based formula retrieval approach discussed in the last chapter is used.
- *Text Indexing and Retrieval*: It retrieves a set of related mathematical documents based on an input text query.
- *Combined Formula and Text Ranking*: It combines and ranks the retrieved mathematical documents obtained from the formula and text query retrievals, and produces the final retrieval results.

In the following sections, we will discuss each process in details.

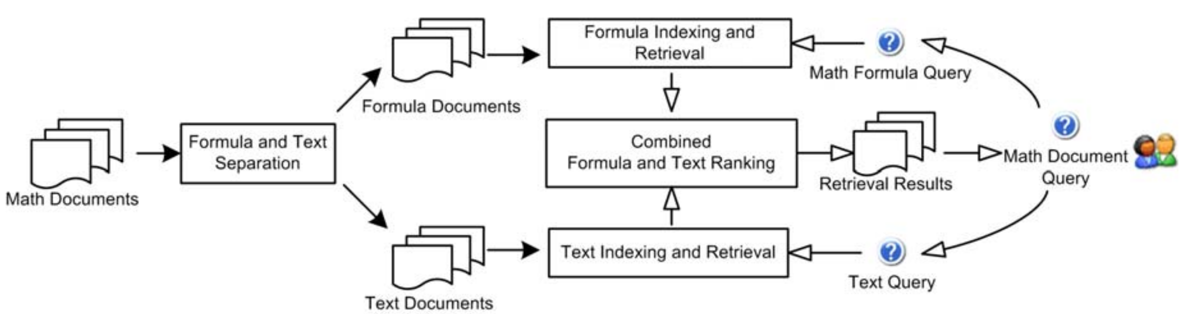


Figure 4.2: Proposed Mathematical Document Retrieval Approach.

4.3 Formula and Text Separation

This process is used to separate the formula and text components from mathematical documents to form individual formula and text documents. For each mathematical document, the formulas are extracted first as they are encoded in *AsciiMath* and enclosed by the character symbol '‛', e.g., '2sinxcosx'. The remaining contents are then treated as text document.

4.4 Formula Indexing and Retrieval

For retrieving mathematical formulas, we use the index-based formula retrieval approach discussed in Section 3.5. The approach only retrieves results for a single formula query. However, a mathematical document may contain more than one formula. To tackle this problem, we process each formula retrieval separately. For example, if a mathematical document query contains n formulas, after formula retrieval, n lists of retrieval results corresponding to each formula query are obtained. Next, we combine these lists of results together to form the final list of retrieval results. During this process, the ranking score for the formula which occurs in different lists of retrieval results is updated by taking the average of its matching scores in each retrieval result list.

If a mathematical document query has a set of formulas $QF(qf_1, qf_2, ..., qf_n)$. After formula retrieval, the lists of retrieval results $L(l_1, l_2, ..., l_n)$ are obtained, in which each list (e.g., l_1, l_2, etc.) corresponds to each formula query (e.g., qf_1, qf_2, etc.). The combined formula score for mathematical document D is obtained based on its contained formulas ($f \in D$) as follows:

$$Score_{formula}(D, QF) = \frac{\sum_{f \in D} \sum_{qf \in QF, f \in l, l \in L} ||Score_{formula}(f, qf)||_{norm}}{N} \tag{4.1}$$

where N is the number of formulas in D.

For example, in Figure 4.3, if the mathematical document query has three formulas as $QF(qf_1$, qf_2 and $qf_3)$. After formula retrieval, three lists of results are obtained as $L(l_1$, l_2 and $l_3)$. From the retrieval results, if mathematical document D contains three formulas f_1, f_2 and f_3, where f_3 occurs in l_3, and f_1 occurs in both l_1 and l_2. The combined formula score for D is given as follows:

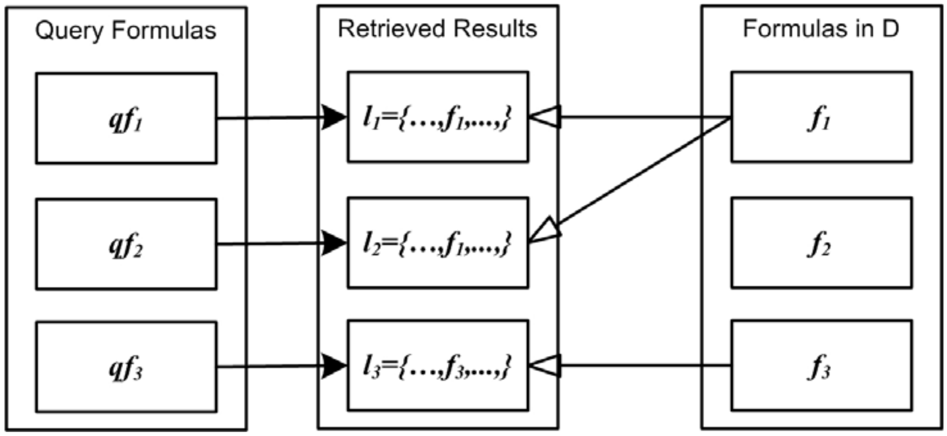


Figure 4.3: Example for the Combined Formula Ranking Process.

$$
\begin{aligned}
Score_{formula}(D, QF) = & \frac{1}{3} \times (||Score_{formula}(f_1, qf_1)||_{norm} + \\
& ||Score_{formula}(f_1, qf_2)||_{norm} + ||Score_{formula}(f_3, qf_3))||_{norm})
\end{aligned}
$$

4.5 Text Indexing and Retrieval

For text retrieval, we implement both the index-based retrieval and clustering-based retrieval techniques. In this section, we only discuss the index-based retrieval technique, as the clustering-based retrieval technique is similar to that discussed in Section 3.6.

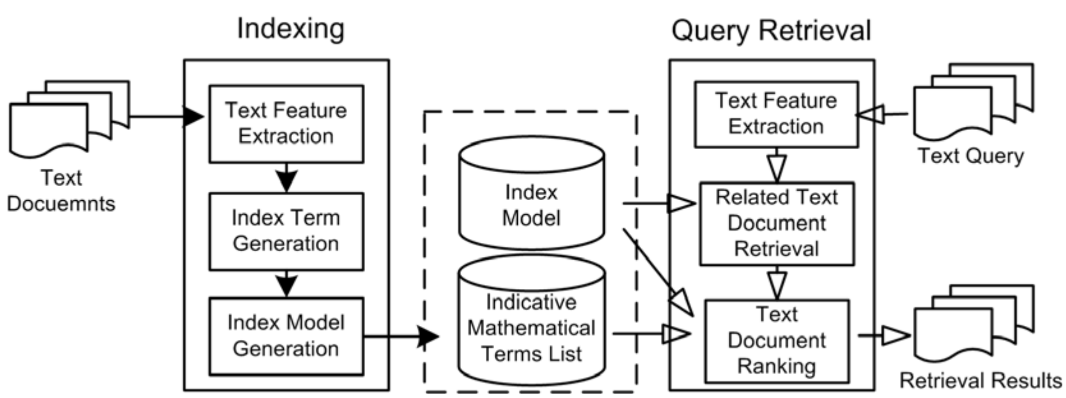


Figure 4.4: Index-Based Text Retrieval Technique.

Figure 4.4 presents the index-based text retrieval technique which consists of the following two processes:

- *Indexing*: It processes the text documents which are separated from the mathematical documents, and creates the index model accordingly. The Indexing process consists of two steps: Text Feature Extraction and Index Terms Generation.

- *Query Retrieval*: Similar to the Query Retrieval process discussed in Formula Retrieval, this process retrieves a list of related documents based on their relevance to a given query. In addition, we have predefined a list of indicative terms to give more weights to mathematical terms. This process contains the following three steps: Text Feature Extraction, Related Text Document Retrieval and Text Document Ranking.

Moreover, in this research, we conduct the performance evaluation of our proposed approach based on the mathematical documents obtained from Gaokao examinations, in which all text documents are written in Chinese. Therefore, text processing techniques for Chinese characters such as word segmentation are also investigated.

4.5.1 Text Feature Extraction

This step extracts text feature terms from the input text documents using common text processing techniques. It consists of the following two steps:

- *Word Segmentation*: It segments Chinese text strings into a sequence of Chinese words. In this research, we have used the word segmentation technique from ICTCLAS (Institute of Computing Technology, Chinese Lexical Analysis System) [159]. In addition, unimportant data such as formatting data and punctuations are filtered out.

- *Stop Words Removal*: It eliminates all insignificant words by adopting the 434 Chinese stop words provided by ICTCLAS. As a result of this step, a list of text terms which represents the text features is obtained.

4.5.2 Index Terms Generation

In this process, the text feature terms obtained from the Text Feature Extraction step are used to generate the index terms. The process is similar to the same step in formula retrieval, in which the n-gram technique, where $n = 1, ..., 4$, is applied.

4.5.3 Index Model Generation

In this step, all the generated index terms are stored as inverted indexes. Figure 4.5 shows an example of this step. In Figure 4.5, given three sample text documents numbered as document

Document ID	Document Contents
1	满足条件的复数在复平面上对应点的轨迹是
2	已知满足下列选项中一定成立的是
3	从长度分别为的五条线段中，任取三条的不同取法共有多少种

⇩ Text Feature Extraction

Document ID	Text Features
1	满足 条件 复数 复 平面 ...
2	已知 满足 下列 选项 ...
3	长度 分别 五条 线段 ...

⇩ Index Term Generation

Document ID	1-gram	2-gram	3-gram	4-gram
1	满足 条件 复数 复 平面 ...	满足$条件 ...	满足$条件$复数 复 ...	满足$条件$复数$复...
2	已知 满足 下列 选项 ...	已知$满足 ...	已知$满足$下列 ...	已知$满足$下列 选项$ ...
3	长度 分别 五条 线段 ...	长度$分别 ...	长度$分别$五条 ...	长度$分别$五条$线段 ...

⇩ Created Indexes

Index Term	Document ID List		
满足	Document 1	Document 2	...
满足$条件	Document 1	...	...
已知$满足$下列	Document 2	...	...
...	...	...	...

Figure 4.5: An Example on Index Model Generation.

1, document 2 and document 3, we first process each document by the Text Feature Extraction process. After that, the corresponding feature terms for each document are obtained. Note that in Figure 4.5, we only present four feature terms for each document for illustration purpose. Then, using the n-gram technique, the index terms from 1-gram to 4-gram are generated. Here, we have used the symbol "$" to separate each term. Finally, the indexes for all sample documents are created.

4.5.4 Related Text Document Retrieval

This process retrieves a set of related text documents according to an input query which is carried out in a similar manner to that in formula retrieval.

4.5.5 Text Document Ranking

In this process, a matching score for each of the retrieved text documents is computed. Then, all the retrieved documents are ranked based on their matching scores.

Mathematics is a specialized field which has many mathematical terms. These terms are indicative in determining the semantic meaning of mathematical documents compared with other commonly used terms. We have identified a list of indicative mathematical terms by analyzing Gaokao's mathematical documents. Table 4.1 shows some examples of indicative mathematical terms. The complete list of indicative mathematical terms is given in Appendix A.

Table 4.1: Some Examples of Indicative Mathematical Terms.

中垂线		互补		俯视图
中线		仰角		偶函数
二次函数		余差		充分条件
二面角		余弦		共轭
减区间		切线		切点
前 n 项和		割线		双曲线
右焦点		回归		周期函数
圆心角		坐标轴		奇偶性

Text Matching Score

The text matching score is computed using the TFIDF scheme [160], which assigns a weight to text feature term t in document d as follows:

$$\mathrm{TFIDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t) \tag{4.2}$$

where $\mathrm{TF}(t,d)$ is the Term Frequency, i.e., the number of occurrences of term t in document d. The $\mathrm{IDF}(t)$ is the Inverse Document Frequency.

Based on the TFIDF scheme, the matching score obtained for each retrieved text document can be computed through the following two steps:

1. If any retrieved text documents contain a query's text feature terms, we add the TFIDF values of the matched terms to its matching score. In this step, the text matching score for text document d to query q is obtained as follows:

$$\sum_{t_j \in S_{MT}(d,q)} \mathrm{TFIDF}(t_j, d) \tag{4.3}$$

   where S_{MT} is the set of matched text feature terms.

2. If any retrieved text documents contain a query's indicative mathematical terms, we add not only the TFIDF values, but also the maximum IDF value of the collection, i.e., $logN$, where N is the number of text documents in the collection, to its text feature

matching score. In this step, the text matching score for text document d to query q is given as follows:

$$\sum_{t_j \in S_{IT}(d,q)} (\text{TFIDF}(t_j, d) + |S_{IT}(d, q)| \times logN) \tag{4.4}$$

where S_{IT} is the set of matched indicative mathematical terms and N is the number of text documents in the collection.

In summary, the total text matching score for text document d to query q is given as follows:

$$Score_{text}(d, q) = \sum_{t_j \in S_{MT}(d,q)} \text{TFIDF}(t_j, d) + \sum_{t_j \in S_{IT}(d,q)} (\text{TFIDF}(t_j, d) + |S_{IT}(d, q)| \times logN) \tag{4.5}$$

where S_{MT} is the set of matched feature terms, S_{IT} is the set of matched indicative terms and N is the number of text documents in the collection.

Text Matching Score Normalization

In this step, we normalize the text matching score obtained by each retrieved text document. To do this, we have employed *byte size normalization* [161]. The normalized weight for document d is obtained as follows:

$$CharLength^{-\alpha}(d), \quad \alpha < 1 \tag{4.6}$$

where $CharLength$ is the number of characters in document d and α is a weighting parameter. According to [161], the value for α is set to 0.75 in this research.

As such, the total text feature matching score for text document d to query q is given as follows:

$$||Score_{text}(d, q)||_{norm} = Score_{text}(d, q) \times CharLength^{-\alpha}(d), \quad \alpha < 1 \tag{4.7}$$

where $Score_{text}(q, d)$ is the original text matching score obtained by d, $CharLength$ is the number of characters in document d and α is the weighting parameter.

4.6 Combined Formula and Text Document Ranking

This step ranks the combined formula and text retrieval results. For computing the ranking score for text documents, we use the text matching score. In addition, the contributions of the formula matching score and text matching score to the final ranking score should not be treated as equal. Therefore, we add a parameter β to adjust the relative importance.

If mathematical document query Q has a set of formulas $QF(qf_1, qf_2, ..., qf_n)$ and text document QT. The text matching score for mathematical document D is given as follows:

$$Score_{text}(D, QT) = ||Score_{text}(D, QT)||_{norm} \tag{4.8}$$

Thus, the combined formula and text score for mathematical document D is obtained as follows:

$$Score(D, Q) = \beta \times Score_{formula}(D, QF) + (1 - \beta) \times Score_{text}(D, QT) \text{ (where } 0 \leq \beta \leq 1) \tag{4.9}$$

where β is the parameter to adjust the relative importance of the formula and text matching scores. The parameter β will be determined experimentally (to be discussed in Section 4.7.2).

4.7 Performance Evaluation

This section presents the performance of the proposed approach for mathematical document retrieval as well as the text retrieval approach.

4.7.1 Dataset

To evaluate the performance of the proposed mathematical document retrieval approach, we have created the Gaokao Mathematical Document Dataset using past Gaokao mathematics examination questions. In the dataset, there are a total of 1898 mathematical questions, which can be categorized into 22 topics. Table 4.2 shows the topics and the total number of questions for each topic. In Table 4.3, it gives some sample mathematical questions.

Table 4.2: Gaokao Mathematical Document Dataset.

Topic ID	Topic Names	Total	Topic ID	Topic Names	Total
1	Function	327	12	Binomial Theorem	57
2	Conic Section	226	13	Equation of Circle and Straight Line	57
3	Solid Geometry	200	14	Straight Line and Plane	55
4	Series	135	15	Application Problem	50
5	Probability	119	16	Triangles	44
6	Set Theory	114	17	Limits	36
7	Vector	99	18	Differentiation	24
8	Inequality	88	19	Mathematical Induction	23
9	Complex Number	78	20	Statistics	19
10	Trigonometry	74	21	Elementary Algorithm	8
11	Permutation and Combination	60	22	Parametric Equation and Polar Coordinates	5

Table 4.3: Sample Mathematical Questions.

Sample Math Questions	Topics
设全集是实数集 R，$M=\{x\mid -2\le x\le 2\}$，$N=\{x\mid x<1\}$，则 $\overline{M}\cap N$ 等于	Set Theory
函数 $f(x)=x^2-2ax-3$ 在区间 $[1,2]$ 上存在反函数的充分必要条件是	Function
设 $\{a_n\}$ 是公差为正数的等差数列，若 $a_1+a_2+a_3=15$，$a_1a_2a_3=80$，则 $a_{11}+a_{12}+a_{13}=?$	Series
已知 a、b、c 满足 $c<b<a$，且 $ac<0$，那么下列选项中一定成立的是	Inequality
函数 $f(x)=\cos 2x-2\sqrt{3}\sin x\cos x$ 的最小正周期是	Trigonometry
从原点向圆 $x^2+y^2-12y+27=0$ 作两条切线，则该圆夹在两条切线间的劣弧长为	Equation of Circle and Straight Line
过抛物线 $y^2=2px(p>0)$ 上一定点 $P(x_0,y_0)(y_0>0)$，作两条直线分别交抛物线于 $A(x_1,y_1)$，$B(x_2,y_2)$ (I) 求该抛物线上纵坐标为 $\frac{p}{2}$ 的点到其焦点 F 的距离 (II) 当 PA 与 PB 的斜率存在且倾斜角互补时，求 $\frac{(y_1+y_2)}{y_0}$ 的值，并证明直线 AB 的斜率是非零常数	Conic Section
已知 A, B, C 三点在球心为 O，半径为 R 的球面上，$AC\perp BC$，且 $AB=R$，那么 A,B 两点的球面距离为?，球心到平面 ABC 的距离为?.	Solid Geometry
平面 α 的斜线 AB 交 α 于点 B，过定点 A 的动直线 l 与 AB 垂直，且交 α 于点 C，则动点 C 的轨迹是	Straight Line and Plane

4.7.2 Experiments on Text Retrieval

In this section, we evaluate the performance of the text retrieval approach. To conduct the experiment on the text retrieval approach, we extract only the text contents from the Goakao Mathematical Document Dataset to form the Gaokao Text Dataset. We then divide the Gaokao Text Dataset into training and testing sets. As there are only 8 and 5 questions in the last two topics of the Gaokao Mathematical Document Dataset as shown in Table

4.2, we only use the text documents from the first 20 topics for the experiment. We select 60 text documents with 3 documents from each topic for testing, while the remaining 1825 documents are used for training. Then, we further divide the testing data into two sets: development set and test set. The development set contains 30 documents which are used to tune the parameters for the K-means and AHC clustering algorithms. The test set contains the other 30 documents for performance evaluation using the tuned parameters. Similar to the evaluation measures for formula retrieval, we also use Precision at 5 (P@5), Precision at 10 (P@10) and Mean Average Precision (MAP) to evaluate the performance of the text retrieval approach. Table 4.4 summaries the number of text documents used for training and testing.

Table 4.4: Number of Text Documents Used for Training and Testing.

	Training Data	**Testing Data**		**Total**
		Development Set	***Test Set***	
Number of Formulas	1825	30	30	1885

Parameter Tuning

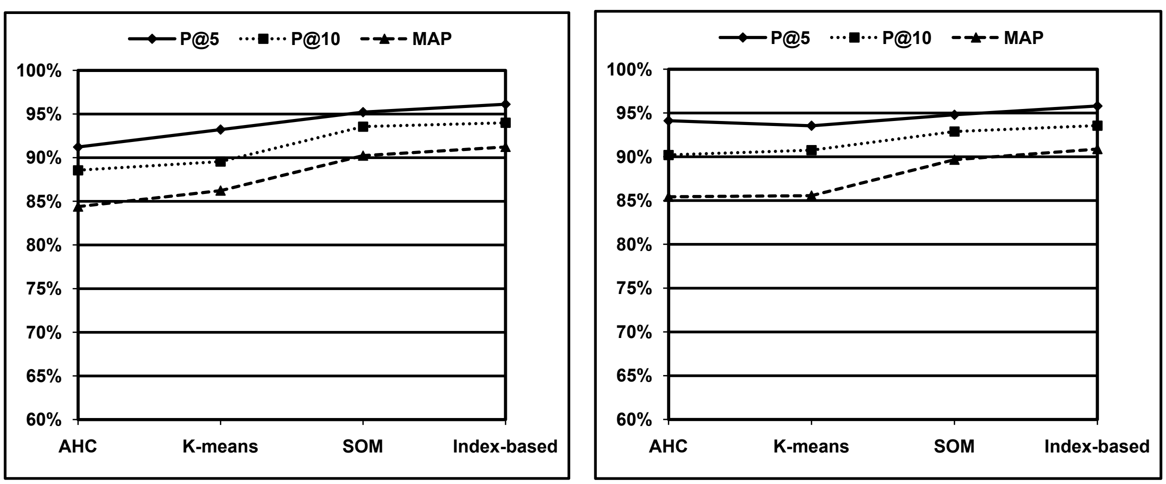
In this section, we tune the parameters for the clustering-based retrieval technique. For the clustering-based technique, the parameters used for the K-means and AHC algorithms need to be tuned. To do this, we use the similar experimental settings discussed in the same step in Section 3.7.3. After the experiment, we found that the two clustering algorithms have achieved their best MAP performance of 86.23% and 84.39% when the number of centroids K for K-means and the stopping criterion for AHC are set to 10 and 20 respectively. Therefore, the best parameter values for K of K-means and the stopping criterion of AHC are obtained at 10 and 20 respectively.

Performance Results

In this section, we evaluate the performance of the index-based retrieval technique as well as the clustering-based retrieval technique using the text documents in the test set. In the experiment, we use the best parameter values for the K-means and AHC algorithms discussed in parameter tuning. In addition, the performance evaluation based on the development set is also given. Table 4.5 and Figure 4.6 present the performance results, which have shown that both techniques for text retrieval have achieved promising performance. Moreover, the index-based technique has achieved better performance than the clustering-based technique. The index-based retrieval technique has achieved high performance of 96.12% and 95.80% for P@5 on the development set and test set respectively. It has shown that relevant documents are generally ranked at the top of the retrieval results. On the other hand, among the three algorithms in the clustering-based technique, SOM has performed better than the other two algorithms on K-means and AHC.

Table 4.5: Performance Results for Text Retrieval.

	Development Set				Test Set		
	P@5	*P@10*	*MAP*		*P@5*	*P@10*	*MAP*
Index-based	96.12%	94.00%	91.44%		95.80%	93.56%	90.88%
K-means	93.21%	89.56%	86.13%		93.54%	90.76%	85.56%
AHC	91.17%	88.42%	84.39%		91.82%	87.89%	84.43%
SOM	95.26%	93.53%	90.76%		94.80%	92.89%	89.67%



(a) Development Set (b) Test Set

Figure 4.6: Performance Comparison for Text Retrieval.

4.7.3 Experiments on Math Document Retrieval

In this section, we evaluate the performance of our proposed mathematical document retrieval approach. To do this, we have applied the same experimental settings used in the performance evaluation for text retrieval. The difference is that we use the Gaokao Mathematical Document Dataset to conduct the experiment instead of using only the text documents.

In addition, as the index-based technique has performed better than the clustering-based technique for both formula retrieval and text retrieval, in this section, we only evaluate the performance of the index-based technique for mathematical document retrieval.

Parameter Tuning

For the index-based retrieval technique, we need to tune the parameter β which adjusts the relative importance of mathematical formula features and text features in the Combined Formula and Text Ranking process (see Equation (4.9)). In order to determine the best value for β, we set β from 0 to 1 in intervals of 0.1. Figure 4.7 shows the performance results of the different values of β based on MAP for the index-based technique. From Figure 4.7, we can see that the index-based technique has achieved better MAP performance when β is set in the range between 0 and 0.5. And the best MAP performance is achieved when β is set to 0.3. Therefore, the best value for β is obtained at 0.3. It means that the text features are more significant than the mathematical formula features when retrieving mathematical documents.

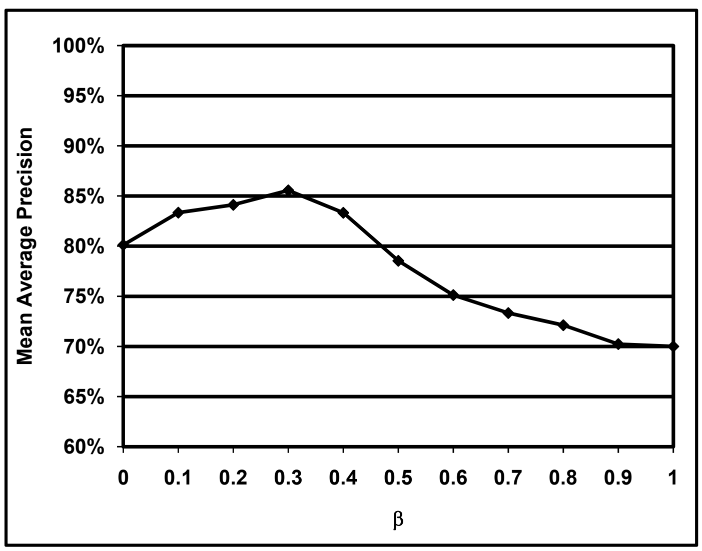


Figure 4.7: Performance Results based on Different Values of β for Math Document Retrieval.

Performance Results

In this section, we evaluate the performance of the proposed mathematical document retrieval approach using the mathematical documents in the test set. We use the tuned parameter β at 0.3 to conduct the experiment. In addition, the performance evaluation based on the development set is also given. Table 4.6 and Figure 4.8 give the performance results of the proposed approach which is promising. For the test set, the performance results for the index-based retrieval technique are 94.21%, 91.88% and 85.12% for P@5, P@10 and MAP respectively.

Table 4.6: Performance Results for Math Document Retrieval.

	P@5	P@10	MAP
Development Set	94.54%	92.12%	85.56%
Test Set	94.21%	91.88%	85.12%

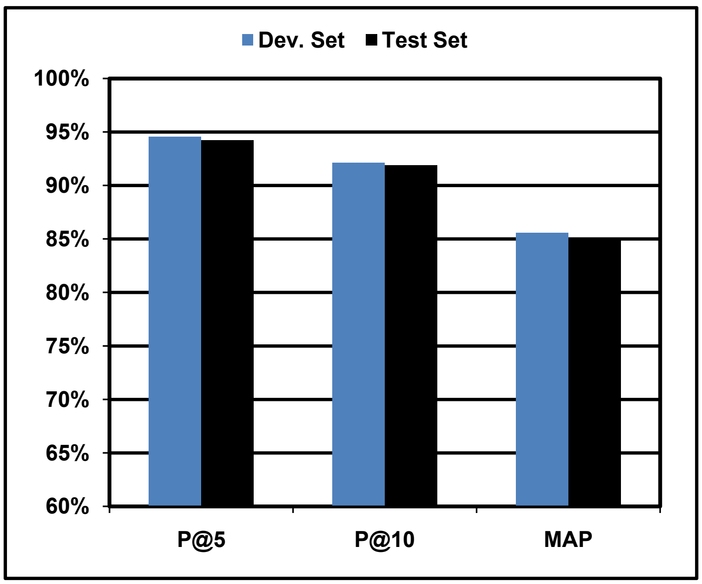


Figure 4.8: Performance Results for Math Document Retrieval.

4.8 Summary

In this chapter, we have presented our proposed mathematical document retrieval approach based on both textual and formula features. In the proposed approach, we first separate textual contents from formulas in mathematical documents for subsequent processing. For text retrieval, we have adopted the index-based approach, which uses the TFIDF weighing scheme with byte size normalization. For formula retrieval, we have used the proposed formula retrieval approach presented in Chapter 3. When retrieving mathematical documents, both text retrieval results and formula retrieval results are combined and ranked. In addition, the performance of the proposed index-based approach is also evaluated. The performance results have shown that the proposed index-based technique has achieved promising performance.

Chapter 5

Mathematical Topic Classification

Mathematical topic classification aims to help users to categorize their posted mathematical questions into appropriate topics automatically in the MathQA Community. It identifies the topics of an input mathematical question query. In this chapter, we propose an approach for automatic topic classification of mathematical questions based on Support Vector Machine (SVM). The proposed SVM-based topic classification approach and its performance evaluation in comparison with other classical classification techniques will be presented.

5.1 Mathematical Topic Classification in MathQA Community

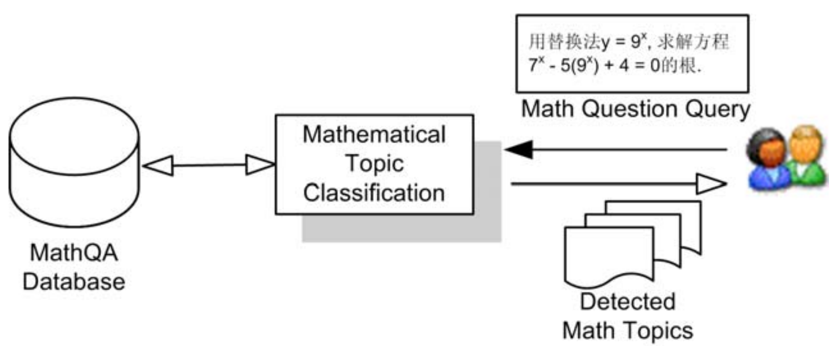


Figure 5.1: Mathematical Question Topic Classification in MathQA Community.

Figure 5.1 shows the mathematical topic classification process in the MathQA Community. First, an input mathematical question document is entered by a user as a query. Then, based on the proposed SVM-based topic classification approach, it identifies the topics that the question query belongs to. Finally, the detected topics of the question query are returned to the user.

5.2 Proposed Approach

Figure 5.2 shows the proposed approach for mathematical question topic classification which consists of the following two processes:

- *Training*: It trains the topic classification models using different classification algorithms. This process comprises the following three steps: Feature Extraction, Transformation and Classification Model Generation. When the Training process is completed, the Topic Classification Models will be generated and stored for the subsequent Classification process.
- *Classification*: It detects the possible topics of an input question query based on the generated Topic Classification Models. The Classification process comprises three steps: Preprocessing, Query Transformation and Query Topic Detection.

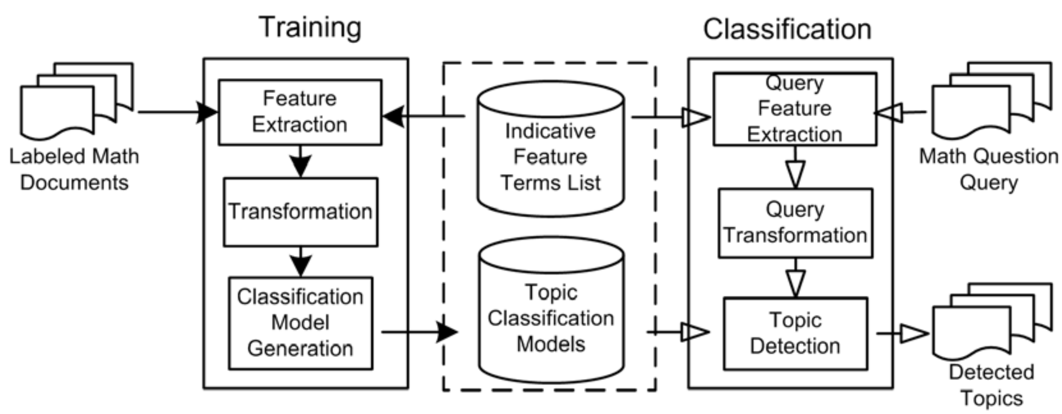


Figure 5.2: Proposed Mathematical Topic Classification Approach.

In the following sections, we will discuss each of the processes in details.

5.3 Feature Extraction

In this step, each topic-labeled mathematical question document is processed into formula and text feature terms. Figure 5.3 shows the Feature Extraction process.

First, the Formula and Text Separation step separates formula and text components from the mathematical question document to form individual formula and text documents. This step is the same as that in Chapter 4. Then, Preprocessing is applied to extract both text and formula features from both types of documents. Preprocessing is divided into two steps:

- *Text Preprocessing*; and
- *Formula Preprocessing.*

In addition, in order to extract key feature terms, the Indicative Feature Terms Matching step is employed.

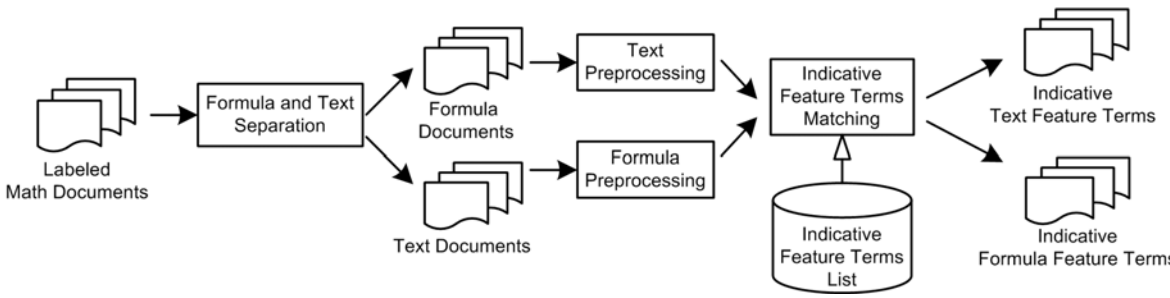


Figure 5.3: Feature Extraction Process.

5.3.1 Text Preprocessing

This step is similar to that of the Text Feature Extraction step in Section 4.5.1. Both the Word Segmentation and Stop Words Removal processes are used for Text Preprocessing. As a result of this step, a list of text feature terms is obtained.

5.3.2 Formula Preprocessing

Due to the existence of both semantic and structural information, the preprocessing step for mathematical formulas is more complex than that of text documents. For mathematical topic classification purpose, the formula features should be representative enough to reflect the underlying characteristics for each mathematical topic. Here, we have used the four types of formula features as discussed in Section 3.4. The only difference is that the real values of the constant features of important operators such as *power* and *subscript* are kept. This is because the actual values are highly important in reflecting the unique formula patterns for certain topics. For example, both formulas $y = kx \pm b$ (line equation) and $x^2 + y^2 = 1$ (circle equation) are frequently occurred in the topic of *Equation of circle and straight line*. Thus, the formula patterns for this topic are $y^1 = x^1 \pm C$ and $x^2 + y^2 = \pm C$, where C is a constant.

In this case, the real values for the *power* function should be retained. For example, *msup*\$1 and *msup*\$2 are used to represent the constants 1 and 2 respectively.

5.3.3 Indicative Feature Terms Matching

As a result of both Text and Formula Preprocessing steps, we have obtained a list of text and formula feature terms. However, most of them are too general to be utilized for the classification process. Therefore, we identify a list of indicative feature terms to match the key mathematical features. The list contains two types of feature terms:

- *Indicative Text Feature Terms*: It is constructed by using the indicative mathematical terms discussed in the Text Document Ranking process (see Appendix A).
- *Indicative Formula Feature Terms*: It is constructed by carefully selecting a set of most representative formula feature terms extracted from the formula collection (see Appendix B). Table 5.1 shows some sample indicative formula feature terms.

Table 5.1: Sample Indicative Formula Feature Terms.

Formula Feature Terms	Meaning	Sample Formulas
msubt	variable with power of t	x^t, y^t
msub2	variable with power of 2	x^2, y^2
γ	Gamma (Unicode)	3γ
msub7	variable with subscript of 7	$a_7 + 1$
log	logarithmic function	$\log x, \log(x^2 + 1)$

5.4 Transformation

In this step, we convert the indicative text and formula feature terms obtained from the Feature Extraction step into the corresponding vector representations, which are then used for generating the classification model. To do this, we have used the list of indicative feature terms discussed in the last section. The indicative feature terms list contains a total of 830 distinct feature terms, including 565 text feature terms and 265 formula feature terms. Based on this list, we then map both types of feature terms into a 1×830 vector and the value in each dimension is weighed by TFIDF. In addition, to perform the Classification Model Generation process, it is required to assign a topic label to each input vector. Thus, we add another dimension at the end of each transformed vector to store the topic label, i.e., topic ID. Therefore, the input mathematical documents are transformed into their labeled vector representations. Finally, we feed these mathematical document vectors into the classification algorithms for training. Note that, different from the same step in the clustering-based formula retrieval approach, we do not normalize the obtained TFIDF vectors during the classification process. This is because we found that using the normalized vectors to perform the classification process greatly reduces the accuracy after several experiments. It is probably caused by the fact that the classification algorithms may not perform well while the values of the input vectors do not show evidenced differences (all values are between 0 and 1 after normalization).

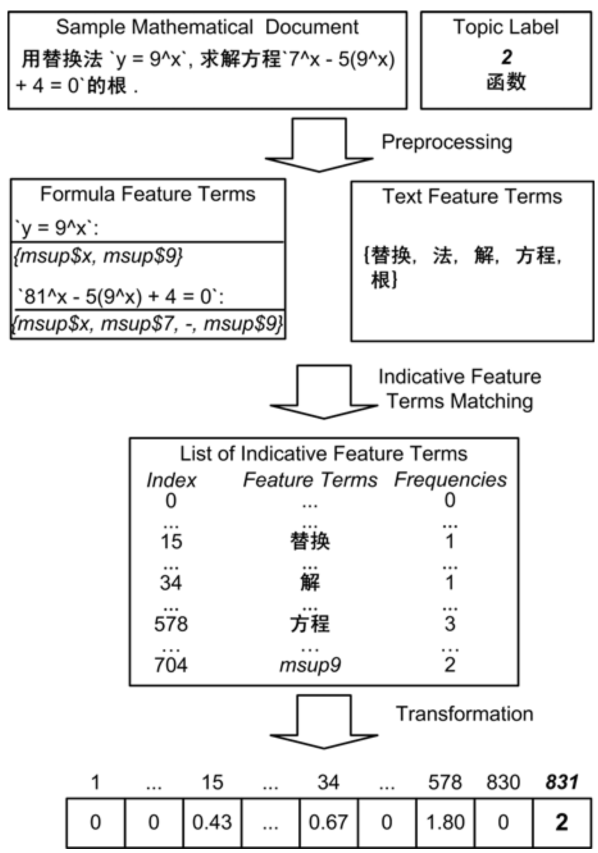


Figure 5.4: An Example on Transformation.

Figure 5.4 illustrates an example on the Transformation process for a sample mathematical document.

5.5 Classification Model Generation

Based on the transformed mathematical document feature vectors, this step aims to generate Topic Classification Models for topic detection. As Support Vector Machine (SVM) [96] has been shown to perform very well on text classification tasks, in which data is represented in a relatively high dimensional space using sparse feature vectors [162, 163, 164], we will apply SVM to achieve the mathematical topic classification task.

SVM is a popular machine learning method for many Natural Language Processing (NLP) tasks. It searches for a maximum margin separating hyper plane between two different classes in the training dataset, measured by the sum of the distances from the hyper plane to the closest positive and negative correctly classified samples. Data points that are found to lie at the edge of the optimal hyper plane are called *support vectors* as illustrated in Figure 5.5. Unlike other algorithms, SVM only considers these support vectors in generating the classification model rather than the entire training data set. So the size of the training set is not usually an issue.

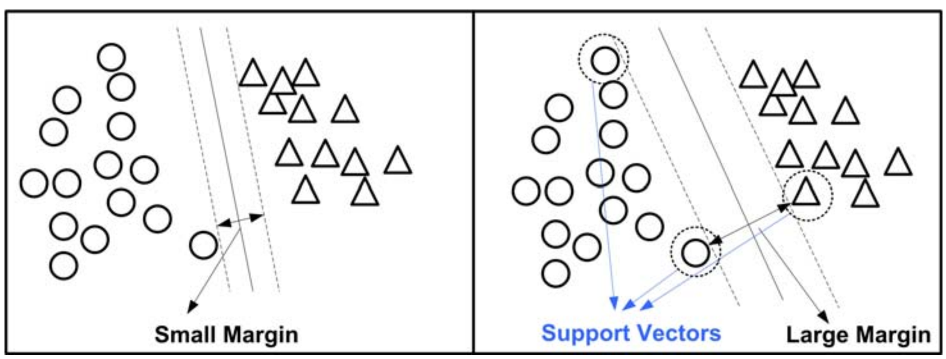


Figure 5.5: Support Vectors in SVM.

To construct an optimal hyper plane, SVM employs an iterative training algorithm to minimize an error function. According to the form of the error function, SVM models fall into the following four groups [165]: *C-SVM classification*, *nu-SVM classification*, *epsilon-SVM regression* and *nu-SVM regression*. Table 5.2 summaries the error functions and the corresponding constraints for these four groups of SVM models. In addition, $\phi(x)$ in the constraints is often called the *kernel function*, which includes *Linear*, *Polynomial*, *Radial Basis Function* (*RBF*) and *Sigmoid* [166].

As mentioned, we formulate the mathematical question topic classification problem as a multi-class classification problem and use a SVM classifier [167, 168] for the task. However,

Table 5.2: Error Functions and Constraints for SVM.

Type of SVM	Error Function	Constraints
C-SVM Classification	$\frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i$	(1) $y_i(w^T\phi(x_i)+b) \geq 1-\xi_i$ (2) $\xi_i \geq 0, i=1,\dots,N$
nu-SVM Classification	$\frac{1}{2}w^T w - v\rho + \frac{1}{N}\sum_{i=1}^{N}\xi_i$	(1) $y_i(w^T\phi(x_i)+b) \geq \rho-\xi_i$ (2) $\xi_i \geq 0, i=1,\dots,N$ (3) $\rho \geq 0$
epsion-SVM Regression	$\frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i v\rho + C\sum_{i=1}^{N}\xi_i'$	(1) $y_i(w^T\phi(x_i)+b-y_i) \geq \epsilon + \xi_i'$ (2) $y_i-(w^T\phi(x_i)-b_i) \geq \epsilon+\xi_i$ (3) $\xi_i, \xi_i' \geq 0, i=1,\dots,N$
nu-SVM Regression	$\frac{1}{2}w^T w - C(v+\frac{1}{N})\sum_{i=1}^{N}(\xi_i+\xi_i')$	(1) $w^T\phi(x_i)+b-y_i \geq \epsilon+\xi_i'$ (2) $y_i-(w^T\phi(x_i)-b_i) \geq \epsilon+\xi_i$ (3) $\xi_i, \xi_i' \geq 0, i=1,\dots,N$

SVM is a binary classifier. There are two common approaches for extending SVM to multi-class classification problems [169]. The first is known as the *pairwise* approach, where a separate binary classifier is trained for each of the class pairs and their outputs are combined to predict the classes. This approach requires the training of $\frac{N(N-1)}{N}$ binary classifiers, where N is the number of different classes. The second, known as *one* vs *all* (*ova*) approach, involves training n classifiers for a n-class problem. The classifiers are trained to discriminate between examples of each class, and those belonging to all other classes.

As reported in [169] that the *ova* approach outperforms the *pairwise* approach, we use the *ova* classification approach for the proposed approach. While for performance comparison, we will use another two classical text classification algorithms, namely Naïve Bayes (NB) [94] and k-Nearest Neighbor (k-NN) [95], which are commonly applied for the document classification task [170, 171, 172, 173]. In addition, as the C-SVM (the error function is $\frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i$) with the RBF kernel function ($exp(-\gamma||x_i - x_j||^2)$) achieves better performance in text classification [169, 174, 175], in this research, we use the C-SVM with the RBF kernel function to configure the SVM topic classifier.

Figure 5.6 shows the Classification Model Generation process. As shown in Figure 5.6, given a set of mathematical document vectors, for each topic, a SVM binary classifier is then trained. For each topic, the mathematical documents belong to the topic are labeled as *positive* samples, and those that belong to other topics are labeled as *negative* samples. As a result, n SVM classifiers for n topics are trained.

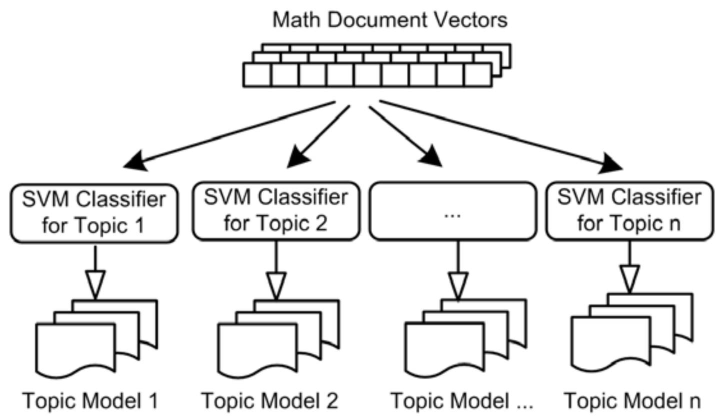


Figure 5.6: Classification Model Generation.

5.6 Topic Classification Models

The Topic Classification Models store all the necessary classification information for each trained SVM classifier, which could be used for detecting the topics for an input question query. As discussed in the last section, one SVM classifier is generated for each topic, there will be a total of n Topic Classification Models generated, where n is the number of topics. The detailed information for SVM models can be found in [176].

5.7 Query Feature Extraction and Query Transformation

In this step, an input question query will be processed in the similar manner as that of both the Feature Extraction and Transformation processes in the Training process. As a result, the input question query will be transformed into a TFIDF vector.

5.8 Topic Detection

In this step, the generated TFIDF query vector is fed into each of the generated Topic Classification Models to detect the possible topic labels for the query. As there are n topic classification models according to the *ova* approach, it needs n classification processes to complete the topic detection process for a query (where n is the total number of topics). Figure 5.7 illustrates the Topic Detection process.

As shown in Figure 5.7, in order to detect the possible topics for a given mathematical question query, the query vector will go through each topic classification model. As a result, one label will be obtained for each model which indicates whether this question belongs to

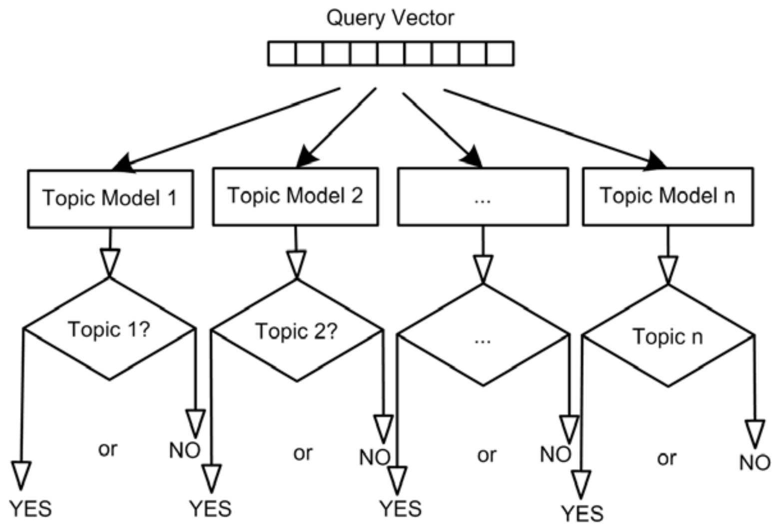


Figure 5.7: Question Topic Detection.

that topic or not. Finally, we gather all the output topics having the YES labels together as the detected topics for the question query.

5.9 Performance Evaluation

In this section, we evaluate the performance of the proposed mathematical topic classification approach.

5.9.1 Evaluation Measures

To evaluate the classification performance, it is important to define the evaluation criteria precisely. Most evaluation techniques start with a confusion matrix or contingency table [177]. Table 5.3 shows an example confusion matrix. Based on the confusion matrix, the most popular metrics for classification performance evaluation are *accuracy* and *error rate*. They are defined as follows:

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN} = \frac{a + d}{a + b + c + d} \tag{5.1}$$

$$error\_rate = 1 - accuracy \tag{5.2}$$

In this research, we will use *accuracy* for performance evaluation. However, the *accuracy* metric is sometimes misleading or meaningless for highly unbalanced data. For example,

Table 5.3: Confusion Matrix.

		Predicted Class	
		Class = Yes	Class = No
Actual Class	Class = Yes	a TP (True Positive)	b FN (False Negative)
	Class = No	c FP (False Positive)	d TN (True Negative)

consider the following binary classification problem:

- Class 0 - number of examples = 9990
- Class 1 - number of examples = 10

If the model predicts everything to be class 0, then the *accuracy* will be equal to 99.9%, i.e., $accuracy = \frac{9990}{10000} = 99.9\%$. In order to avoid this problem, we have also employed *precision*, *recall* and *F-measure* (F1) to evaluate the classification performance. Based on the confusion matrix given in Table 5.3, the three metrics are defined as follows:

$$precision = \frac{TP}{TP + FP} = \frac{a}{a + c} \tag{5.3}$$

$$recall = \frac{TP}{TP + FN} = \frac{a}{a + b} \tag{5.4}$$

$$F - measure\ (F1) = \frac{2pr}{p + r} = \frac{2a}{2a + b + c} \tag{5.5}$$

As such, the performance evaluation for the proposed mathematical document classification approach will be measured by the following four metrics: *accuracy*, *precision*, *recall* and *F-measure* (F1).

5.9.2 Experiments

The Gaokao Mathematical Document Dataset discussed in Section 4.7.1 is used for conducting the experiment. It has a total of 22 mathematical topics. However, as some topics such as *Elementary Algorithm* and *Differentiation* have limited number of questions, only 17 topics are selected for conducting the experiment. Table 5.4 shows the 17 topics and the total number of mathematical questions each topic contains. We use the 10-fold cross validation [178, 179, 180] to evaluate the performance for the proposed SVM-based topic classifier. In addition, we also evaluate the performance of another two classical topic classification algorithms, namely Naïve Bayes (NB) and k-Nearest Neighbor (k-NN), for performance comparison.

Table 5.4: Topics and Number of Math Documents for Performance Evaluation.

Topic ID	Topic Name	Total	Topic ID	Topic Name	Total
1	Function	327	10	Trigonometry	74
2	Conic Section	226	11	Permutation and combination	60
3	Solid Geometry	200	12	Binomial Theorem	57
4	Series	135	13	Equation of Circle and Straight Line	57
5	Probability	119	14	Straight Line and Plane	55
6	Set Theory	114	15	Application Problem	50
7	Vector	99	16	Triangles	44
8	Inequality	88	17	Limits	36
9	Complex Number	78			

Evaluation Features

In the experiment, we have used the following three ways of feature selection for evaluating mathematical question topic classification:

- *All Text Features* (ATF): Only text features are extracted from the mathematical question documents. It has a total of 2,230 distinct text feature terms.

- *Indicative Text Features* (ITF): Indicative text features are extracted from the mathematical question documents. They are text feature terms contained in the list of indicative text feature terms. As such, indicative text feature terms are a subset of text feature terms. It has a total of 565 text feature terms (see Appendix A).

- *Indicative Text and Formula Features* (ITFF): Indicative text and formula feature terms are extracted from the mathematical question documents according to the lists of indicative text and formula terms. It has a total of 565 text feature terms and 265 formula feature terms.

Table 5.5: Performance Results based on All Text Features.

Topic IDs	**Accuracy**			**Precision**			**Recall**			**F-Measure (F1)**		
	SVM	**NB**	**k-NN**	**SVM**	**NB**	**k-NN**	**SVM**	**NB**	**k-NN**	**SVM**	**NB**	**k-NN**
1	**93.3**	69.3	90.3	**93.1**	87.0	90.2	**93.3**	69.3	90.3	**93.1**	73.1	90.2
2	93.7	93.1	**94.1**	93.8	**94.8**	93.8	93.7	93.1	**94.1**	92.8	93.6	**93.8**
3	95.1	**95.8**	95.2	95.2	**96.8**	95.0	95.1	**95.8**	95.2	94.5	96.1	**94.8**
4	**97.3**	73.8	96.3	**97.2**	93.6	96.2	**97.3**	73.8	96.3	**97.1**	80.1	96.2
5	94.2	92.8	**95.5**	**94.5**	**95.4**	95.3	94.2	92.8	**95.5**	91.7	93.7	**94.5**
6	**96.9**	72.7	95.3	**96.9**	94.3	94.8	**96.6**	72.7	94.8	**96.5**	79.9	95.0
7	**97.3**	61.2	96.5	**97.2**	94.6	96.1	**97.3**	61.2	96.5	**96.9**	71.5	96.2
8	**97.9**	66.5	97.2	**97.9**	95.7	97.1	**97.9**	66.5	97.2	**97.6**	76.1	97.1
9	**99.3**	73.2	99.1	**99.3**	96.4	99.0	**99.3**	73.2	99.1	**99.2**	81.3	99.0
10	**96.7**	68.3	95.2	**96.6**	96.1	95.5	**96.7**	68.3	95.2	**95.6**	77.9	95.3
11	97.2	93.9	**97.4**	97.2	**97.5**	97.2	97.2	93.9	**97.4**	96.0	95.2	**96.5**
12	**99.6**	77.5	99.0	**99.6**	97.3	99.0	**99.6**	77.5	99.0	**99.6**	84.9	99.0
13	**97.9**	83.6	96.6	**97.8**	96.9	96.5	**97.9**	83.6	96.6	**97.4**	88.8	96.6
14	94.9	94.9	**97.4**	**97.7**	**97.7**	97.1	94.9	94.9	**97.4**	95.9	95.9	**97.2**
15	**97.4**	92.7	**97.4**	94.8	**97.6**	94.8	**97.4**	92.7	**97.4**	**96.1**	94.6	**96.1**
16	**98.3**	68.5	97.5	**98.2**	97.4	97.1	**98.3**	68.5	97.5	**97.9**	79.3	97.3
17	**97.3**	69.8	95.2	**97.2**	93.2	95.2	**97.3**	63.2	95.2	**96.9**	72.6	95.2
Avg.	**96.7**	79.3	96.2	**96.7**	95.4	95.9	**96.7**	78.9	96.2	**96.2**	84.4	95.9

Table 5.6: Performance Results based on Indicative Text Features.

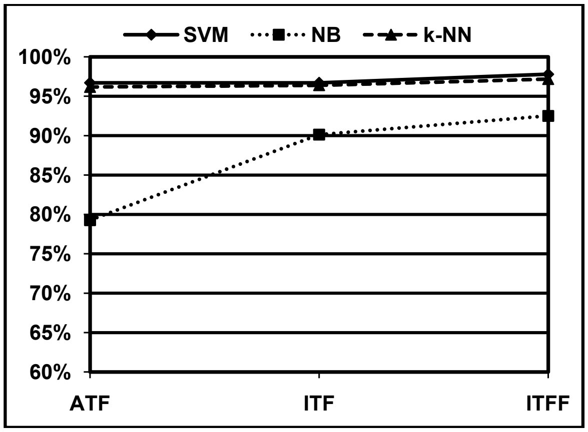
Topic IDs	**Accuracy**			**Precision**			**Recall**			**F-Measure (F1)**		
	SVM	**NB**	**k-NN**	**SVM**	**NB**	**k-NN**	**SVM**	**NB**	**k-NN**	**SVM**	**NB**	**k-NN**
1	**92.5**	77.4	90.7	**92.3**	89.2	90.7	**92.5**	77.4	90.7	**92.1**	80.1	90.7
2	92.3	92.9	**94.8**	92.5	94.3	**94.6**	92.3	92.9	**94.8**	90.8	93.3	**94.5**
3	92.6	**95.8**	95.7	92.7	**96.5**	95.6	92.6	**95.8**	95.7	90.9	**96.0**	95.4
4	96.4	95.9	**97.0**	96.8	96.9	**97.0**	96.9	95.9	**97.0**	96.7	96.2	**97.0**
5	94.9	95.9	**97.3**	**94.8**	97.1	97.1	94.9	95.9	**97.3**	93.4	96.4	**97.2**
6	**96.8**	90.0	96.1	**96.7**	92.9	95.7	**96.8**	90.0	96.1	**96.4**	91.3	95.8
7	**96.9**	84.2	96.1	**96.8**	95.2	95.9	**96.9**	84.2	96.1	**96.5**	88.2	96.0
8	**97.9**	90.9	97.3	**97.9**	96.4	97.2	**97.9**	90.9	97.3	**97.6**	92.8	97.2
9	**99.2**	86.4	98.8	**99.2**	96.7	98.8	**99.2**	86.4	98.8	**99.2**	90.1	98.8
10	**96.6**	74.9	90.7	**96.4**	95.7	90.7	**96.6**	74.9	90.7	**95.3**	82.6	90.7
11	97.9	93.6	**98.3**	98.0	97.6	**98.4**	**97.9**	93.6	98.3	**97.5**	95.1	98.3
12	99.2	98.7	**99.4**	99.2	98.9	**99.4**	**99.2**	98.7	99.4	**99.2**	98.8	99.4
13	**98.2**	93.0	96.3	**98.1**	96.6	96.5	**98.2**	93.1	96.5	**97.9**	94.5	96.4
14	**98.2**	96.0	97.6	**98.2**	97.8	97.4	**98.2**	96.0	97.6	**97.8**	96.7	97.5
15	**97.5**	94.8	97.4	**97.5**	96.4	94.8	**97.5**	94.8	97.4	**96.3**	95.4	96.1
16	**98.1**	84.8	97.4	**97.9**	97.4	96.9	**98.1**	84.8	97.4	**97.6**	89.9	97.1
17	**98.1**	86.9	97.8	96.2	98.0	96.6	**98.1**	86.9	97.8	**97.2**	91.5	97.1
Avg.	**96.7**	90.1	96.4	**96.5**	96.1	96.1	**96.7**	90.1	96.4	96.0	92.3	**96.2**

Table 5.7: Performance Results based on Indicative Text And Formula Features.

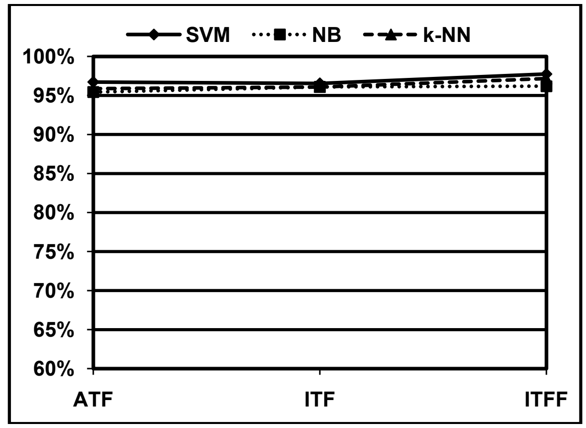
Topic IDs	Accuracy			Precision			Recall			F-Measure (F1)		
	SVM	NB	k-NN	SVM	NB	k-NN	SVM	NB	k-NN	SVM	NB	k-NN
1	**95.2**	91.0	92.0	**95.1**	93.1	91.8	**95.2**	91.1	92.1	**95.1**	93.6	91.8
2	**96.4**	91.9	95.0	**96.5**	94.1	94.9	**96.2**	91.9	95.0	**96.4**	92.5	94.6
3	**97.2**	90.8	96.4	**97.2**	93.9	96.2	**97.2**	90.8	96.4	**97.1**	92.3	96.3
4	**97.8**	94.8	97.3	**97.7**	96.4	94.7	**97.8**	94.8	97.5	**97.7**	95.3	97.3
5	97.7	95.4	**98.2**	97.7	95.5	**98.1**	97.7	95.5	**98.2**	97.6	95.5	**98.1**
6	**97.0**	93.3	97.0	**97.0**	94.9	96.9	**97.0**	93.3	**97.0**	96.6	93.9	**96.7**
7	97.9	93.4	**98.1**	97.8	96.0	**98.0**	97.9	93.4	**98.1**	97.7	94.3	**98.0**
8	**97.7**	89.7	97.2	**97.7**	96.2	97.1	**97.7**	89.7	97.2	**97.7**	92.0	97.1
9	**99.8**	88.6	97.9	**99.8**	96.6	98.6	**99.8**	88.6	97.9	**99.8**	91.4	98.1
10	**97.6**	89.7	96.7	**97.5**	96.2	96.8	**97.6**	89.7	96.7	**97.2**	92.0	96.8
11	**98.3**	88.3	97.8	**98.2**	97.0	98.1	**98.3**	88.3	97.8	**98.1**	91.6	97.9
12	**99.5**	95.0	**99.5**	**99.5**	97.9	**99.5**	**99.5**	95.1	**99.5**	**99.5**	96.1	**99.5**
13	**98.0**	87.6	96.0	**97.8**	96.6	96.0	**98.0**	87.6	96.0	**97.7**	91.2	96.4
14	97.3	94.7	**94.6**	97.4	**97.6**	97.4	97.3	94.7	**97.6**	96.3	95.8	**97.5**
15	**97.4**	96.9	**97.4**	**97.5**	96.9	**97.5**	**97.4**	96.2	**97.4**	96.2	**96.4**	96.2
16	98.0	94.7	**98.2**	97.7	97.6	**98.0**	98.1	94.7	**98.2**	97.6	95.8	**98.1**
17	**99.5**	97.0	99.2	**99.5**	98.7	99.4	**99.5**	97.0	99.2	**99.5**	97.6	99.3
Avg.	**97.8**	92.5	97.0	**97.7**	96.2	97.0	**97.8**	92.5	97.2	**97.5**	94.0	97.0

Performance Results

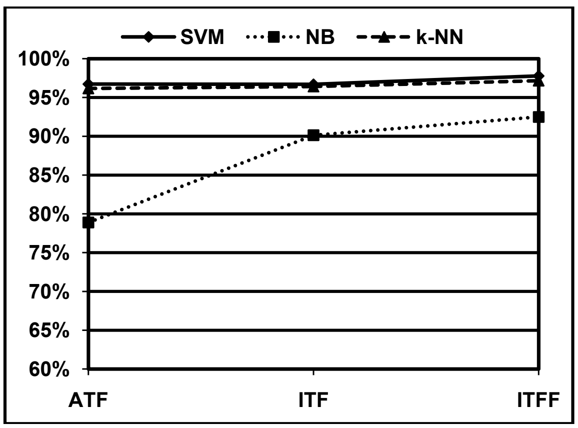
In this section, we present the performance results of the proposed SVM-based topic classifier in comparison with the other two classification algorithms, namely NB and k-NN, based on each topic. Table 5.5, Table 5.6 and Table 5.7 give the performance results for each topic according to the different ways of feature selection. Figure 5.8 gives the performance results for each classification technique in terms of average accuracy, precision, recall and F-measure based on the different ways of feature selection. As can be seen, SVM has consistently outperformed NB and k-NN in most topics based on the different ways of feature selection. Also, k-NN has performed better than NB for most topics in terms of the four measures on accuracy, precision, recall and F-measure. In addition, the performance of the SVM-based classifier that uses indicative text and formula features has performed better than using the other two ways of feature selection. Based on using indicative text and formula features, the SVM-based classifier has achieved considerably high accuracy (97.8%), precision (97.7%), recall (97.8%) and F-measure (97.5%) for topic classification.



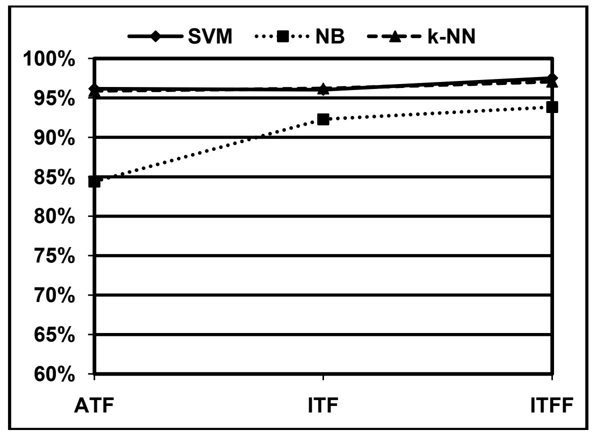
(a) accuracy



(b) precision



(c) recall



(d) F-measure (F1)

Figure 5.8: Performance Results for SVM, NB and k-NN based on Different Selected Features.

5.10 Summary

In this chapter, we have presented our proposed SVM-based approach for mathematical topic classification. In the proposed approach, we first separate textual contents from formulas in mathematical documents for training. Then, text preprocessing and formula preprocessing are performed for text and mathematical formula feature extraction. Indicative text and formula feature terms are also identified. These indicative terms are transformed and fed into SVM for training the classification models. During classification, the input question query is processed and transformed, and the topics of the input question are then detected with the classification models. In addition, the performance of the proposed SVM-based approach is also evaluated. The performance results have shown that the proposed SVM-based approach has achieved high performance for topic classification and has outperformed the classical classification algorithms such as NB and k-NN.

Chapter 6

Human Expert Finding

As some of the posted mathematical questions are unable to find answerers, human expert finding aims to help users to find experts who are most likely able to answer their posted questions in the MathQA Community. According to an input mathematical question query, the expert finding process retrieves a list of ranked experts based on their past records of answering questions. In this chapter, we propose an approach for expert finding in the MathQA Community. The proposed approach and its performance evaluation will be presented.

6.1 Expert Finding in MathQA Community

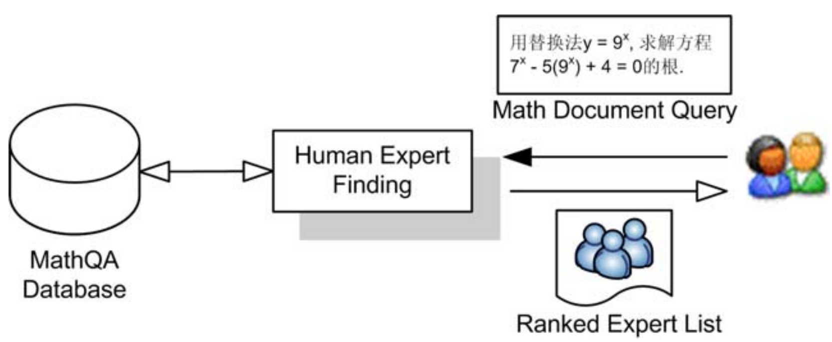


Figure 6.1: Human Expert Finding in MathQA Community.

Figure 6.1 shows the human expert finding process in the MathQA Community. First, an input mathematical question document is entered by a user as a query. Then, based on the proposed expert finding approach, it retrieves a list of experts from the MathQA Database according to the similarity between the textual and mathematical features of past questions answered by users and the input question query. Finally, a ranked list of experts is retrieved and displayed.

6.2 Proposed Approach

Figure 6.2 shows the proposed human expert finding approach. It consists of the following processes:

- *Data Preprocessing*. It preprocesses all the resolved questions stored in the MathQA Database into Question-Answerer (QA) pairs.
- *Query Topic Detection*. It detects all possible topics of an input question query using a topic classifier.
- *Expert Retrieval*. It performs the expert finding process. There are two techniques for expert retrieval: index-based retrieval and Latent Semantic Indexing (LSI) [46]-based retrieval. The index-based retrieval technique is based on the document retrieval technique discussed in Chapter 4.

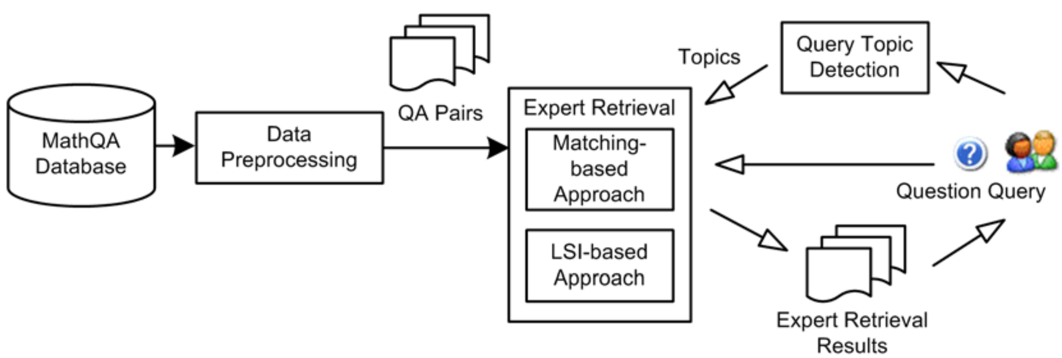


Figure 6.2: Proposed Expert Finding Approach.

In the following sections, we will discuss each process in details.

6.3 Data Preprocessing

Similar to Baidu Zhidao [10] and Yahoo! Answers [9], each record in the MathQA Database contains the following four attributes:

- *Asker*: It contains the identity of the user who has posted the question.
- *Answerer*: It contains the identity of the user who has answered the question.

- *Question*: It contains the question description. According to different answer statuses, each question may fall into one of the following three categories: open question, undecided question (question in voting) and resolved question.

- *Answer*: It contains the answer of the question. There are two types of answers, namely *best answer* and *candidate answer*. In MathQA Community, a question may have more than one candidate answer. However, it contains only one best answer for a resolved question.

Figure 6.3 shows an example of a resolved question in the MathQA Database. In this research, the resolved questions, in which both the question description and the answerers who have posted the best answers are used for finding experts. Different from the open and undecided questions, the resolved questions have the complete body contents including asker, answerers, question and answers. Also, most of these contents are carefully checked by either the asker himself or the system administrator. Similarly, the "best" answer is usually chosen by the asker or voted by others, which ensures its correctness compared with other candidate answers.

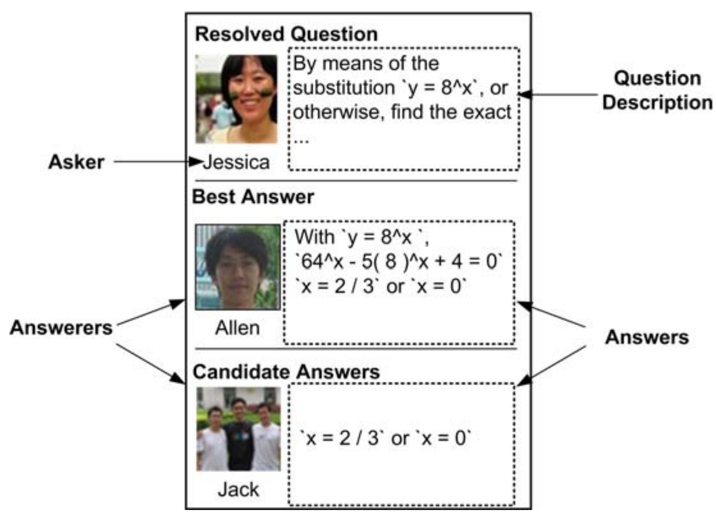


Figure 6.3: An Example of a Resolved Question in the MathQA Database.

As the best answers of the resolved questions are used for finding experts, the question and answerer parts should be extracted from the resolved question records for processing. The question and answerer parts are called Question-Answerer pair (QA pair). In Data Preprocessing, for each topic, we process all resolved questions stored in the MathQA Database into QA pairs. Figure 6.4 shows an example of the QA pair for the resolved question given in Figure 6.3.

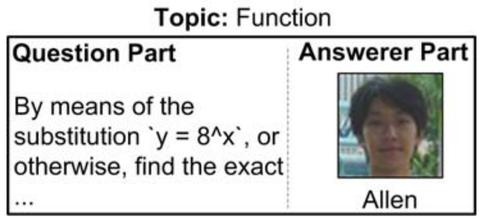


Figure 6.4: An Example QA Pair.

6.4 Query Topic Detection

This process uses a topic classifier to detect the possible topics for an input question query. To achieve this, we have used the proposed SVM-based mathematical topic classification approach discussed in Chapter 5 which gives the best performance in comparison with the other two classical classification techniques on Naïve Bayes (NB) and k-Nearest Neighbor (k-NN).

6.5 Index-based Expert Retrieval

As discussed in Chapter 4, the index-based document retrieval technique has achieved better performance than that of the clustering-based document retrieval technique for mathematical document retrieval. For expert finding and retrieval, we have used the index-based document retrieval technique for computing the similarity between the past questions answered by each user and the input question query. Figure 6.5 shows the index-based expert retrieval technique which consists of the following steps:

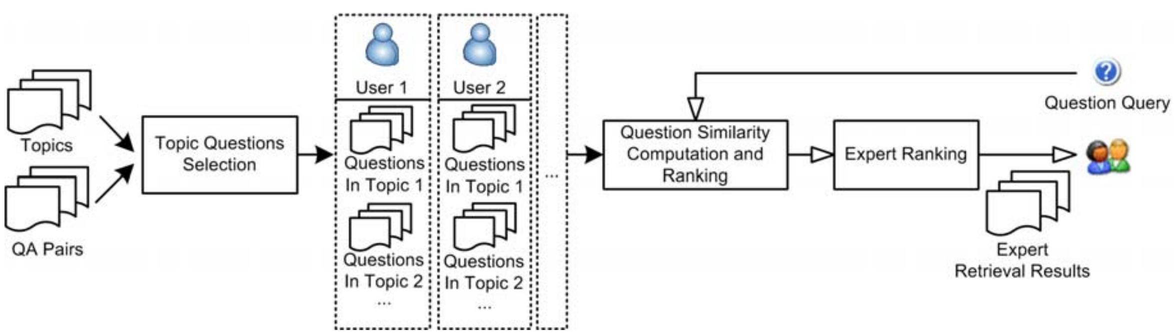


Figure 6.5: Index-based Expert Finding Approach.

- *Topic Questions Selection*: It selects all the questions answered by each user from the detected topics.

- *Question Similarity Computation and Ranking*: It computes and ranks the selected questions according to the question query for each user.
- *Expert Ranking*: It ranks all the users based on the similarity scores of the questions answered by them.

6.5.1 Topic Questions Selection

In this step, based on the input QA pairs and the query's detected topics, we first select all the questions answered by each user (or answerer). As a result, a list of answerers together with their past answered questions from different detected topics are generated. An example is shown in Figure 6.5. However, to be an expert, an answerer should have successfully solved many questions under the detected topics. To filter out unqualified answerers, we select only those answerers who have answered at least 5 questions successfully in order for further processing.

6.5.2 Question Similarity Computation and Ranking

In this step, for each answerer, we compute the similarity between his answered questions and the question query according to the textual and mathematical formula features using the index-based mathematical document retrieval technique discussed in Chapter 4. As a result, a similarity score is computed for each question. Finally, we rank all the questions based on the similarity scores.

6.5.3 Expert Ranking

In this step, to find and rank experts, we only consider the scores for top n (e.g., 3) questions in the ranked list for each answerer. In this research, the value of n will be evaluated experimentally and discussed in the section on performance evaluation. Next, we take the average of the similarity scores for the n questions as the *expert score* for each answerer. Finally, all answerers are ranked based on the expert scores.

Figure 6.6 shows an example on the Expert Ranking step. In Figure 6.6, the expert scores for the three users (1, 2 and 3) are computed. As a result, the ranking order for the three users (or answerers) is User 1, User 2 and User 3 with ranking scores of 0.7, 0.5 and 0.2 respectively.

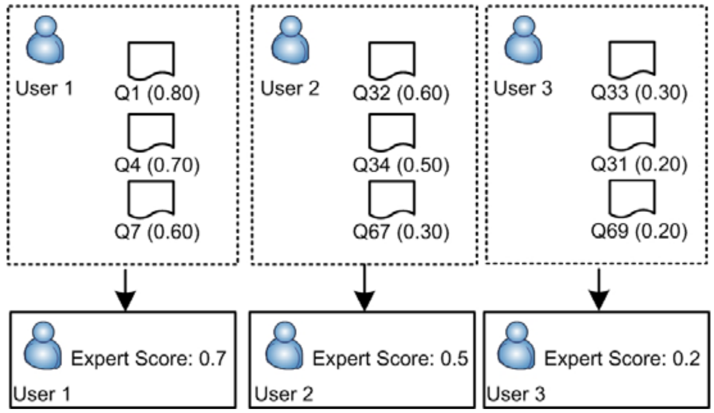


Figure 6.6: An Example on Expert Ranking.

6.6 LSI-based Expert Retrieval

Latent Semantic Indexing (LSI) is a widely used Information Retrieval (IR) technique. Based on the Vector Space Model (VSM) [181], this approach treats all documents as a *term-document matrix* [92]. Usually, the values within the matrix are weighed by TFIDF of each key term. LSI has been developed to overcome problems with synonymy and polysemy that occurred in prior vectorial approaches, thereby improving the basic vector space model by replacing the original term-document matrix with an approximation [92]. And this is often done by using Singular Value Decomposition (SVD) [182, 183]. By using the approximation of term-document matrix, the noise on synonymy and polysemy existing in the original matrix may be reduced.

In this research, we propose a LSI-based technique for expert retrieval. Figure 6.7 shows the proposed LSI-based expert retrieval technique which consists of the following two processes:

- *Model Generation*: It generates the LSI model by using the LSI technique on the feature vectors of the question part of the QA pairs. This process comprises the following four steps: Topic Questions Selection, Feature Extraction and Transformation, LSI Subspace Mapping and LSI Model Generation.

- *Query Retrieval*: It retrieves a list of experts from the generated LSI model based on the input question query. This process consists of the following four steps: Query Feature Extraction and Transformation, Query LSI Subspace Mapping, Question Similarity Computation and Ranking, and Expert Ranking.

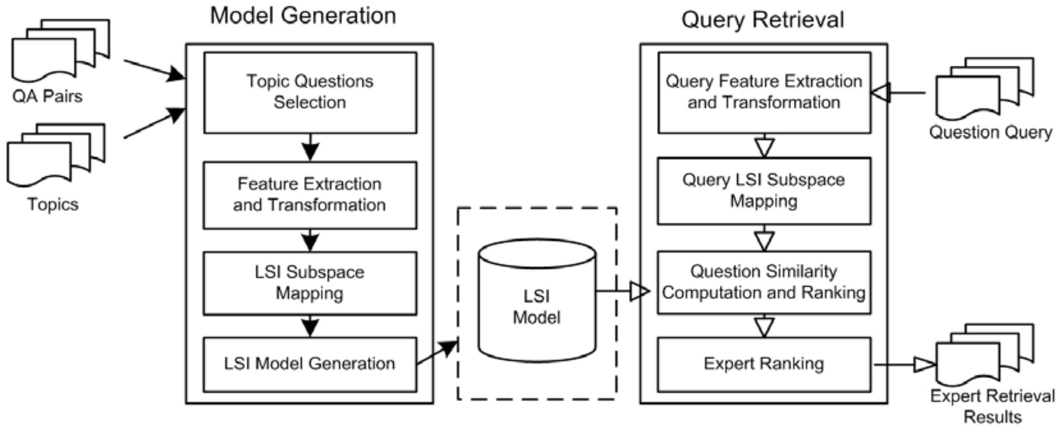


Figure 6.7: LSI-based Expert Retrieval Approach.

6.6.1 Topic Questions Selection

This step is similar to the same step discussed in the index-based expert retrieval technique. It selects a list of answerers with all their past answered questions in each detected topic based on the input QA pairs and the query's detected topics.

6.6.2 Feature Extraction and Transformation

This step uses the Feature Extraction and Transformation processes discussed in Section 5.3 and Section 5.4. The questions answered by each answerer are transformed into mathematical document feature vectors.

6.6.3 LSI Subspace Mapping

Based on the transformed mathematical document feature vectors, this step maps all the feature vectors into a new LSI subspace using the LSI technique. In this step, we first convert all mathematical document feature vectors into a term-document matrix M, which is served as the input for SVD. To construct the matrix M, we transpose the feature vectors and combine them together. Note that one term-document matrix will be constructed for each answerer.

Next, we decompose M into the product of three other matrices. One component matrix U describes the original row entities as vectors of the derived orthogonal factor values, another matrix V describes the original column entities in the same way and the third matrix is a diagonal matrix Σ. The columns of U and V are the left and right singular vectors, and the diagonal matrix is often called the *singular values* of M [92]. Figure 6.8 shows an example

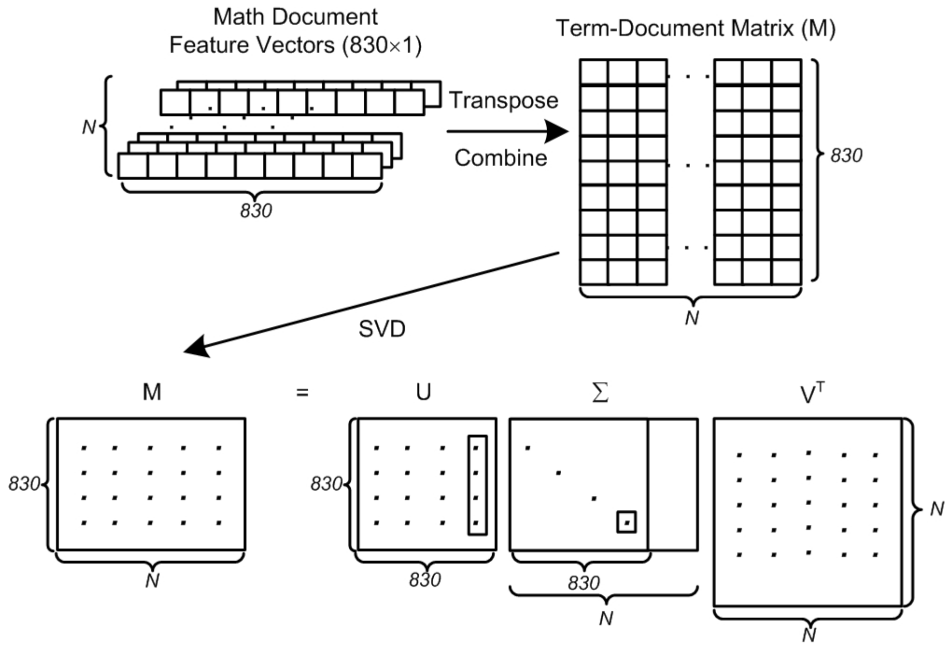


Figure 6.8: Singular Value Decomposition.

on the SVD process. As there are a total of 830 distinct text and formula feature terms (see Section 5.4), the input mathematical document feature vectors have 830 columns which are shown in Figure 6.8. In addition, the number N denotes the total number of mathematical documents.

When we calculate the approximation of the original matrix, the diagonal matrix can be scaled simply by retaining the first k largest diagonal elements to 0, corresponding to the first k columns of the U and V matrices. And this is often called the *k-rank approximation* of M [92]. Figure 6.9 shows an example on the k-rank approximation for the matrix shown in Figure 6.8. According to [92], the reconstruction of the matrix M to obtain its k-rank approximation M_k is given as follows:

$$M_k = U_k \cdot \Sigma_k \cdot V_k^T \tag{6.1}$$

After the LSI subspace is created, the original term-document matrix M will be replaced by the k-rank approximation in the new subspace for subsequent processing.

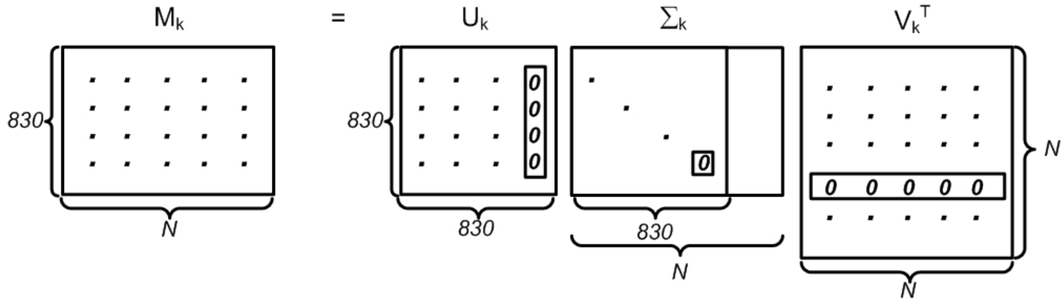


Figure 6.9: The k-Rank Approximation Using SVD.

6.6.4 LSI Model Generation

As each column vector in the matrix M_k corresponds to a k-rank approximation of the original mathematical document feature vector, we have to store M_k into the database. And both matrices U_k and Σ_k are also stored in preparation for the Query LSI Subspace Mapping step. As a result, the LSI model is generated.

6.6.5 Query Feature Extraction and Transformation

In this step, the input question query will be processed in a similar manner to the index-based expert retrieval technique. As a result, the input question query is transformed into a TFIDF query feature vector.

6.6.6 Query LSI Subspace Mapping

This step maps the query feature vector into the LSI subspace based on the generated LSI model. To map the query into the created LSI Subspace, we use the following equation given by [92]:

$$V(\vec{Q}_k) = \Sigma_k^{-1} \cdot U_k^T \cdot V(\vec{Q}) \tag{6.2}$$

where $V(\vec{Q})$ is the query feature vector, and $V(\vec{Q}_k)$ is the k-rank approximation query vector in the LSI subspace.

6.6.7 Question Similarity Computation and Ranking

For each answerer, this step computes and ranks his answered questions according to the input question query using the similar step discussed in the index-based expert retrieval technique. The only difference is that in this step, we use the cosine similarity to measure the similarity between the query feature vector $\vec{V}(Q_k)$ and the mathematical document feature vectors $\vec{V}(D_k)$ of their k-rank approximations. The similarity score is computed as follows:

$$\text{dist}(\vec{V}(Q_k), \vec{V}(D_k)) = \sqrt{\sum_{i=1}^{n} (\vec{V}(q_i) - \vec{V}(d_i))^2} \qquad (6.3)$$

where n is the number of dimensions for the k-rank approximation feature vectors, q_i and d_i are the i^{th}element in the k-rank approximation of the query feature vector and mathematical document feature vector respectively.

6.6.8 Expert Ranking

This step is the same as the same process in the index-based expert retrieval technique. As a result, all answerers are ranked by their expert scores.

6.7 Performance Evaluation

In this section, we evaluate the performance of the proposed human expert finding approach.

6.7.1 Dataset

To evaluate the performance of our proposed human expert finding approach, we have simulated an Expert Dataset from the Gaokao Mathematical Document Dataset discussed in Chapters 4 and 5. The Expert Dataset consists of a total of 888 mathematical questions which are selected from four topics, namely function (327 questions), series (135 questions), conic section (226 questions) and solid geometry (200 questions) that contain most questions compared with other topics. For each mathematical question in the Expert Dataset, the standard answer which is treated as the best answer is also recorded. Then, a total of 20 users are generated as answerers. We divide the total number of questions in the Expert Dataset into two sets: training set and test set. There are 848 questions for the training set and 40 questions (with 10 questions from each topic) for the test set. The questions from the training set are randomly assigned to the users who are then treated as the answerers (i.e., providers of the best answers) to the questions. Table 6.1 shows the number of questions answered by each user based on each topic in the training set. As such, the Question-Answerer pairs for each topic can be generated.

Table 6.1: Number of Questions Answered by Each User based on the Four Topics.

User ID	Solid Geometry	Function	Conic Section	Series
1	3	21	9	6
2	7	18	9	8
3	6	24	7	6
4	10	19	8	5
5	11	18	7	7
6	11	18	16	2
7	13	15	11	8
8	14	13	11	8
9	11	15	13	7
10	11	14	12	5
11	4	10	11	4
12	8	15	13	7
13	6	19	8	2
14	11	13	8	6
15	11	14	10	8
16	12	15	12	8
17	8	16	17	6
18	12	12	15	6
19	9	14	9	6
20	12	14	10	10
Total	**190**	**317**	**216**	**125**

6.7.2 Evaluation Measures

Similar to mathematical formula retrieval and mathematical document retrieval, in this experiment, we evaluate the expert finding approach in terms of both Precision at 5 (P@5) and Mean Average Precision (MAP). In addition, we also use the Mean Reciprocal Rank (MRR) [184, 185, 186] to evaluate the performance of the proposed expert finding approach. MRR aims to evaluate the performance based on whether the expert is ranked at the first position in the retrieved expert list. It is defined as follows:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{6.4}$$

where Q is the set of queries and $rank_i$ is the rank of the first expert for query i.

6.7.3 Experiments

To conduct the experiment, we have used the test set which consists of 10 questions from each topic as test queries. As such, the test set contains a total of 40 questions. For the

LSI-based technique, according to [187], the number of ranks retained for the approximation of the original matrix is set to 20. The parameter β for the index-based technique is set to 0.3.

We have conducted two experiments to evaluate the performance. The first experiment evaluates the Mean Average Precision (MAP) with respect to the number of selected top questions (n) considered in the Expert Ranking step. Based on the number of selected top questions, we vary n from 1 to 20. Figure 6.10 and Figure 6.11 present the experimental results, which have shown that the MAP performance for both index-based and LSI-based expert retrieval techniques increase initially, reach the maximum, and then coming down to a stable value. For the index-based technique, the best MAP performance is obtained at 93.27% when n is set to 8, whereas for the LSI-based technique, the best MAP performance is obtained at 90.00% when n is set to 10.

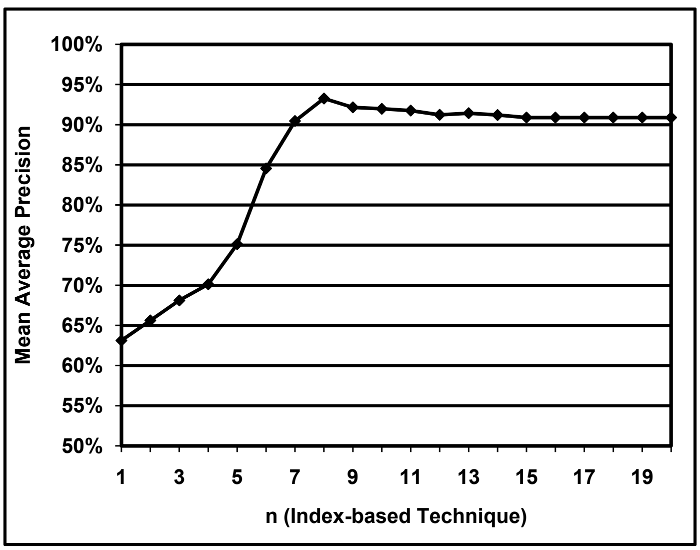


Figure 6.10: Performance Results based on the Value of n for the Index-based Technique.

The second experiment evaluates the performance of the expert finding approach based on each topic. The value of n for the index-based and LSI-based retrieval techniques are set to 8 and 10 respectively. The experimental results are given in Figure 6.12 and Table 6.2, which have shown that both expert retrieval techniques have achieved very good performance for each topic. In addition, the index-based retrieval technique has outperformed the LSI-based retrieval technique on all three measures MRR, P@5 and MAP (see Figure 6.13). In particular, the index-based technique has achieved high performance of 98.75% for MRR, 95.00% for P@5 and 93.27% for MAP.

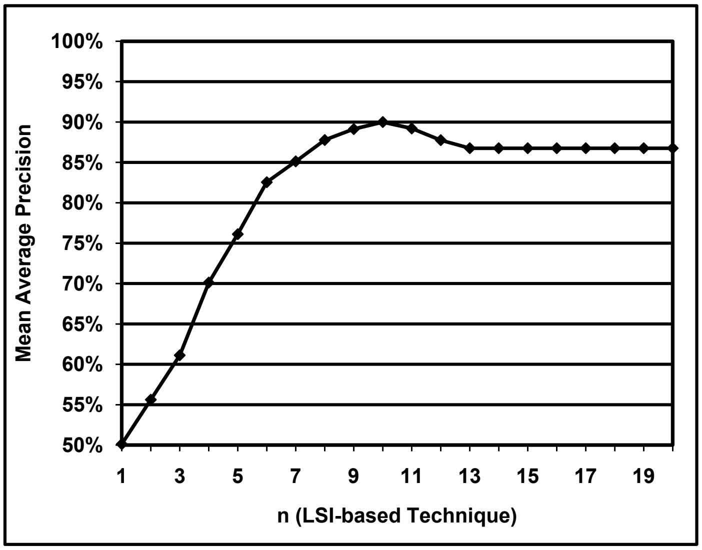
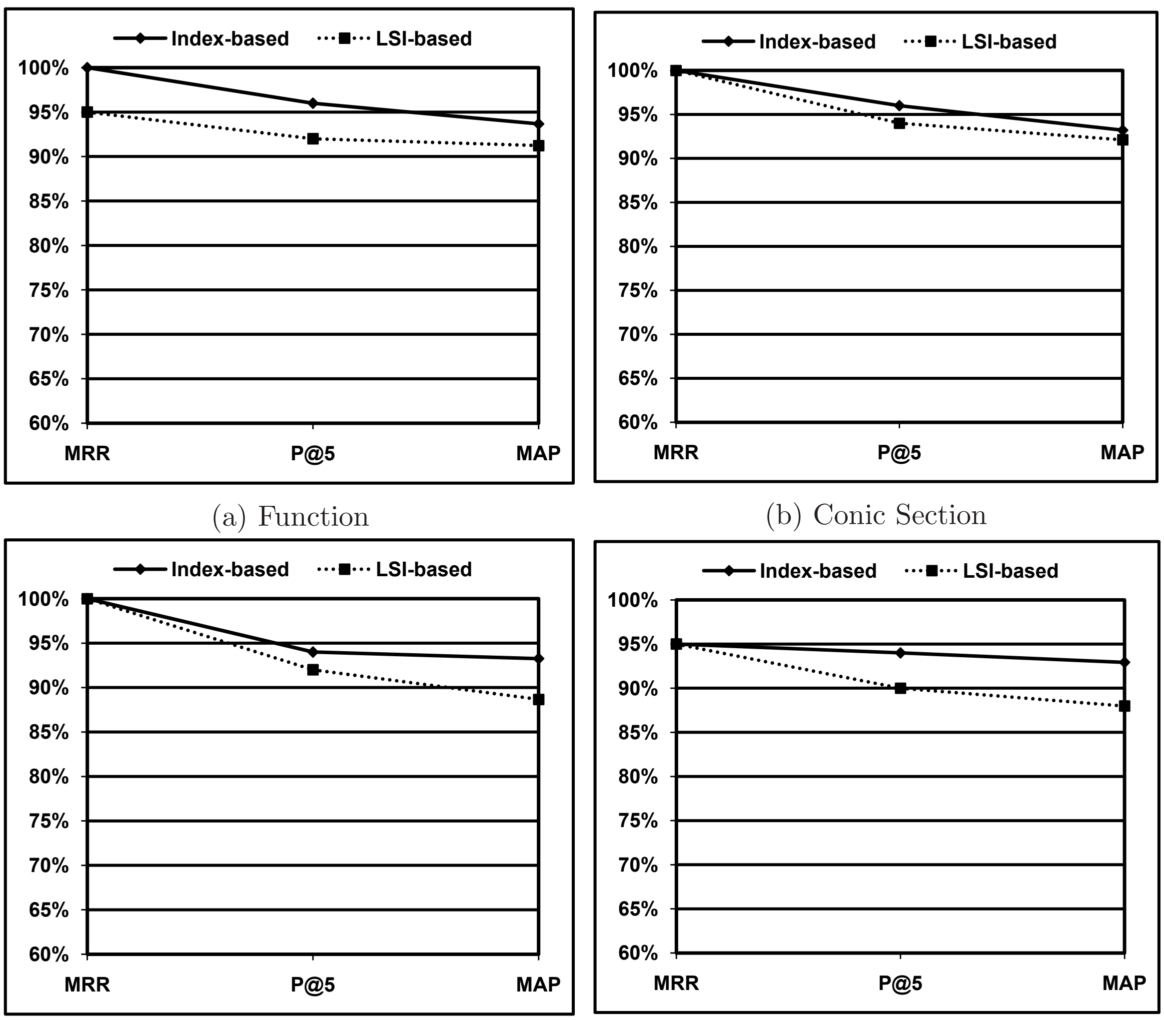


Figure 6.11: Performance Results based on the Value of n for the LSI-based Technique.



(a) Function

(b) Conic Section

(c) Solid Geometry

(d) Series

Figure 6.12: Performance Results based on Topics.

Table 6.2: Performance Results based on Topics.

Topics	Index-based Technique				LSI-based Technique		
	MRR	*P@5*	*MAP*		*MRR*	*P@5*	*MAP*
Function	100.00%	96.00%	93.67%		95.00%	92.00%	91.23%
Conic Section	100.00%	96.00%	93.21%		100.00%	94.00%	92.11%
Solid Geometry	100.00%	94.44%	93.26%		100.00%	92.00%	88.67%
Series	95.00%	94.44%	92.93%		95.00%	90.00%	87.98%
Avg:	98.75%	95.00%	93.27%		97.50%	92.00%	90.00%

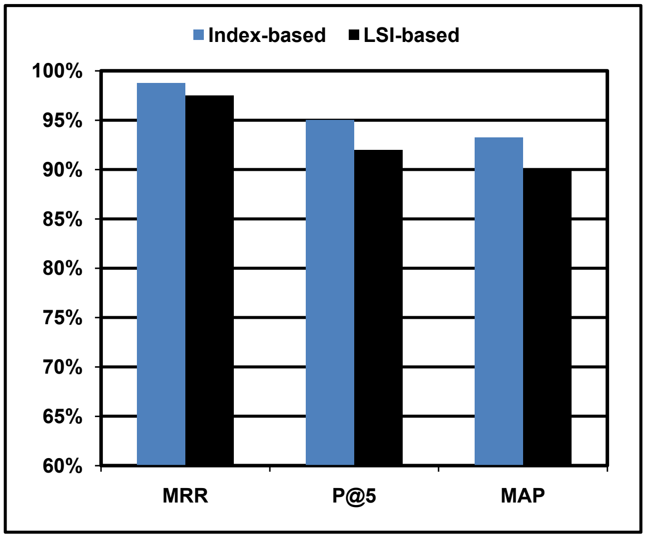


Figure 6.13: Performance Comparison for the Expert Retrieval Techniques.

6.8 Summary

In this chapter, we have presented our proposed human expert finding approach based on both textual and formula features of the past questions answered by each user and the input question query. In the proposed approach, we first extract the Question-Answerer pairs based on the detected topics of the input question query. Then, we implement two expert retrieval techniques, namely index-based retrieval and Latent Semantic Indexing-based retrieval, for retrieving and ranking experts. The index-based retrieval technique uses the index-based document retrieval technique discussed in Chapter 4 for document similarity computation and ranking, whereas the LSI-based retrieval technique uses LSI for the computation and ranking. Moreover, the performance of the proposed approach is also evaluated based on the two retrieval techniques. The performance results have shown that the index-based retrieval technique has outperformed the LSI-based retrieval technique. The index-based retrieval technique has achieved high performance of 98.75% for MRR, 95.00% for P@5 and 93.27%

for MAP.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

For the past few decades, Question Answering (QA) community has been one of the most popular knowledge sharing and problem solving platforms. Instead of referring to the set of related documents retrieved by traditional search engines, users can post their questions to QA communities for finding answers. With QA communities, most questions including those difficult ones could be solved effectively and efficiently. As such, an increasing number of students have joined the educational QA communities for asking help on their homework problems. Although traditional QA communities can handle text-based questions quite well, they are ineffective when dealing with mathematical questions due to the existence of mathematical formula features.

In this research, our main objective is to investigate data mining techniques including clustering and classification for supporting the Mathematical Question Answering (MathQA) Community. To achieve this, we have proposed different techniques for mathematical formula and document retrieval, mathematical topic classification and human expert finding. The proposed techniques are summarized as follows:

- We have proposed an effective formula feature extraction technique [188]. In the proposed technique, four types of formula features, namely semantic, structural, constant and variable are identified and extracted through the following five steps: AsciiMath to MathML Conversion, DOM Tree Construction, Semantic Feature Extraction, Structural Feature Extraction, and Constant and Variable Feature Extraction.

- We have proposed a formula retrieval approach [188]. In the proposed approach, we have investigated two techniques for formula retrieval. The first one is based on the inverted indexing technique and the other is based on the clustering techniques. For the clustering retrieval techniques, three classical clustering algorithms, namely K-means, Agglomerative Hierarchical Clustering (AHC) and Kohonen's Self-Organizing Map (SOM) are implemented. The performance of the proposed approach has been evaluated based on both the index-based and clustering-based techniques.

- We have proposed a mathematical document retrieval approach. The proposed approach is based on the proposed index-based formula retrieval technique and index-based text retrieval technique. The performance of the proposed mathematical document retrieval approach has been evaluated and presented.

- We have proposed a mathematical question topic classification approach. The proposed approach is based on Support Vector Machine (SVM) for topic detection. The performance of the proposed approach has been evaluated in comparison with two other text classification algorithms, namely Naïve Bayes (NB) and k-Nearest Neighbor (k-NN).

- We have proposed an expert finding approach. The proposed approach is based on the proposed mathematical topic classification approach for topic classification. In addition, two techniques have been implemented for expert retrieval. The first one is based on the proposed index-based mathematical document retrieval approach, and the other is based on the Latent Semantic Indexing (LSI) technique. The performance of the proposed expert finding approach has been evaluated based on both the index-based and LSI-based techniques.

As a result, we have incorporated the proposed techniques into the MathQA Community which is currently operational over the Web.

7.2 Future Work

This research has investigated different data mining techniques for enhancing the essential features for supporting the MathQA Community. As a result, the MathQA Community has been developed to provide various mathematical question answering functions to users. For future work, this research can be further extended in the following directions: diagram and graph search, cross language mathematical document retrieval, visualization for retrieval results, extending the MathQA Community for mathematical learning and conducting usability study.

7.2.1 Diagram and Graph Search

In this research, we mainly focus on retrieving mathematical question documents using formulas and textual contents. However, as mentioned in Chapter 1, a mathematical question may also contain diagrams and graphs which also provide important semantic information for retrieving related mathematical documents. As such, retrieval techniques for diagrams and graphs could be investigated for future research. Diagram and graph retrieval is a challenging problem as both diagrams and graphs contain mainly image data instead of textual data. To achieve this, we could conduct the research in a similar manner to that of formula retrieval. First, structural, semantic and spatial features from the diagrams and graphs should be extracted. Image processing techniques can be applied for feature extraction. Then, the extracted features will be represented by its feature vectors. Finally, an index-based technique or clustering-based techniques such as K-means, Self-Organizing Map and Agglomerative Hierarchical Clustering can be investigated for model generation and query retrieval.

7.2.2 Cross Language Mathematical Document Retrieval

Cross Language Information Retrieval (CLIR) has recently attracted much attention from the research community [189, 190, 191, 192, 193]. In this research, we have developed the MathQA Community which supports mathematical document retrieval for related questions. In our experiments, we have used Gaokao's mathematical documents for performance evaluation purpose. As a result, the MathQA Community has been developed based on the Gaokao's mathematical data for supporting mathematical document retrieval. As such, the MathQA Community supports only Chinese question queries. All the user questions and answers are also entered into the MathQA Community in Chinese. As the underlying formats of the mathematical questions are consistent irrespective of which language is being used, it is quite straightforward for the MathQA Community to support mathematical documents from other languages such as the G.C.E. A-Level Mathematics [194].

Therefore, this research could be extended to support CLIR for mathematical documents written in English as well as in Chinese. As such, cross language retrieval for mathematical documents in the MathQA Community could be investigated for future research. This will provide the mathematical search support for users who would like to refer to questions written in another language based on a single question query in a certain language. It will help users to understand the different formats and styles mathematical questions could be set in different countries.

7.2.3 Visualization for Retrieval Results

Currently, the mathematical retrieval results are listed in a ranked order on the display which is not very effective especially when it comes to display mathematical formulas. To improve the presentation of retrieval results, visualization techniques will be useful in order to present the information in a visual manner for better understanding. There are many visualization techniques available including TouchGraph [195], RGraph [196], Word Tree [197], EntityCube [198], Wonder Wheel [199], Timeline [200] and Image Swirl [201] which could be investigated for future research. For example, the TouchGraph could be used to display all the retrieved documents spatially according to the similarity between the retrieved documents and the question query. The RGraph could be used to display the retrieval results, and graph animation will enable users to view the retrieval results dynamically by changing the question query. Word Tree is a visual search tree. It could be used to display the retrieved formulas in a tree-like branching structure. This will enable users to view the retrieved formulas in a connected manner.

7.2.4 Extending the MathQA Community for Mathematical Learning

The MathQA Community contains a rich set of posted questions and answers. This can be served as a mathematical question repository for learning. Therefore, this research could be extended to support mathematical learning. One possible direction for future extension is to support online exercise and test simulation for students. One popular technique for online test simulation is Computerized Adaptive Testing (CAT) [202, 203, 204, 205, 206]. In adaptive testing, the test questions are generated dynamically and attempted by students online. It first selects questions of a suitable difficulty level according to the student's ability, evaluates the responses, and estimates the student's ability according to the responses. Based on some predefined termination conditions, the student's resultant ability or proficiency will then be obtained. The advantage of the adaptive testing approach is that it enables the estimation of a student's ability without the need of requiring the student to attempt all the questions. After that, the student's performance could be monitored and analyzed in order for further performance improvement.

7.2.5 Conducting Usability Study

Currently, the MathQA Community has been implemented and operational. One possible future direction for this research is to conduct a usability study in order to identify whether there are any rooms for further improvement. This could be done as follows. First, we select a group of users (say, 30) for participation. Then, we design a questionnaire to address the different aspects of the MathQA Community such as the system interface, system functional design, user behavior, etc. Finally, we conduct the evaluation study based on the feedback on the questionnaire from the users. During the user evaluation process, a brief introduction and overview about the MathQA Community should be presented. A demonstration on the MathQA Community should also be given to show the functionalities of the MathQA Community. The users should then be given enough time to familiarize themselves with the MathQA Community. In addition, other deployment issues such as scalability, security and stability should also be investigated by conducting online testing and evaluation.

Appendix A

List of Mathematical Terms

1	平面直角坐标系		29	正三角形		57	正三棱柱		85	同一个
2	单调递减区间		30	异面直线		58	前 n 项		86	四面体
3	单调递增区间		31	正态分布		59	二面角		87	增函数
4	最小正周期		32	周期函数		60	三角形		88	对角线
5	反三角函数		33	二次方程		61	抛物线		89	横坐标
6	垂直平分线		34	圆锥曲线		62	双曲线		90	表达式
7	绝对差数列		35	第一象限		63	四棱锥		91	表面积
8	直角三角形		36	第二象限		64	展开式		92	长方体
9	锐角三角形		37	第三象限		65	正周期		93	坐标系
10	钝角三角形		38	第四象限		66	真命题		94	对称轴
11	直二面角		39	当且仅当		67	直棱柱		95	纵坐标
12	非空集合		40	杨辉三角		68	假命题		96	倾斜角
13	正态分布		41	十六进制		69	常数项		97	坐标轴
14	共轭复数		42	同一平面		70	不等式		98	多面体
15	非空子集		43	有且仅有		71	六面体		99	绝对值
16	等差数列		44	有且只有		72	反函数		100	二项式
17	等比数列		45	四则运算		73	定义域		101	公垂线
18	等腰梯形		46	直三棱柱		74	正方体		102	三棱锥
19	等和数列		47	莱布尼茨		75	x0y		103	六边形
20	通项公式		48	解析几何		76	四边形		104	直方图
21	前 n 项和		49	连续函数		77	正方形		105	三视图
22	直角坐标		50	取值范围		78	正整数		106	图象上
23	等比中项		51	最短路线		79	纯虚数		107	多项式
24	三角函数		52	直四棱柱		80	渐近线		108	归纳法
25	充要条件		53	有穷数列		81	上一点		109	轴对称
26	充分条件		54	无穷数列		82	偶函数		110	逆时针
27	必要条件		55	所成的角		83	关系式		111	侧视图
28	有序数对		56	二次函数		84	几何体		112	命中率

113	圆心角		154	增区间		195	y 轴		236	至少
114	平均数		155	奇函数		196	棱长		237	随机
115	平行线		156	减区间		197	棱台		238	顶点
116	无理数		157	极小值		198	性质		239	中点
117	有理数		158	极大值		199	垂足		240	交点
118	流程图		159	奇偶性		200	余差		241	体积
119	立方米		160	集合		201	图象		242	单位
120	解方程		161	真假		202	平面		243	定义
121	逆命题		162	面积		203	非零		244	平行
122	不规则		163	半轴		204	向量		245	数字
123	中垂线		164	实轴		205	焦点		246	数学
124	代数式		165	外接		206	定值		247	相交
125	俯视图		166	共线		207	坐标		248	轨迹
126	准确率		167	最大		208	直线		249	中心
127	分布图		168	虚轴		209	存在		250	公式
128	十进制		169	模型		210	实数		251	半径
129	半圆形		170	证明		211	距离		252	周期
130	反比例		171	长轴		212	交线		253	圆心
131	所成角		172	等价		213	不同		254	复数
132	对称性		173	近似		214	数列		255	夹角
133	小数点		174	服从		215	概率		256	对称
134	不存在		175	最小		216	棱锥		257	斜率
135	平均值		176	递推		217	椭圆		258	理由
136	减函数		177	首项		218	前项		259	甲乙
137	平方米		178	非空		219	命题		260	相等
138	平面图		179	侧棱		220	成立		261	系数
139	期望值		180	棱柱		221	求证		262	组成
140	百分比		181	方程		222	一点		263	结论
141	等分线		182	满足		223	分布		264	说明
142	自变量		183	主值		224	切线		265	下面
143	顺时针		184	极值		225	单调		266	个数
144	蓄水量		185	辐角		226	原点		267	之间
145	离心率		186	一切		227	变量		268	侧面
146	左焦点		187	虚部		228	垂直		269	元素
147	右焦点		188	公和		229	展开		270	公共
148	最小值		189	短轴		230	常数		271	关系
149	最大值		190	动点		231	整数		272	准线
150	三棱柱		191	同向		232	期望		273	判断
151	短半轴		192	对边		233	条件		274	区域
152	长半轴		193	区间		234	正确		275	各项
153	导函数		194	x 轴		235	线段		276	图像

277	大于		318	底面		359	面上		400	自然
278	左右		319	截面		360	两两		401	递减
279	恰好		320	所得		361	以下		402	重心
280	方案		321	抽取		362	偶数		403	顺序
281	方法		322	方向		363	参数		404	乘法
282	球面		323	无穷		364	含有		405	保留
283	直角		324	标准		365	均匀		406	周长
284	相互		325	梯形		366	大致		407	底边
285	相切		326	正数		367	奇数		408	弦长
286	相同		327	独立		368	平分		409	形状
287	矩形		328	球心		369	平均		410	总数
288	结果		329	目标		370	抽样		411	最少
289	解集		330	直径		371	服从		412	最高
290	边长		331	相应		372	次数		413	有序
291	重合		332	相邻		373	正切		414	比值
292	一定		333	等式		374	正弦		415	相距
293	上述		334	等腰		375	比较		416	表面
294	不等		335	等边		376	空间		417	连接
295	互相		336	约束		377	端点		418	通项
296	任选		337	统计		378	精确		419	钝角
297	位于		338	编号		379	给定		420	一半
298	位置		339	菱形		380	至多		421	上面
299	余弦		340	虚数		381	边界		422	两地
300	值域		341	规则		382	选择		423	乙地
301	假设		342	规定		383	递增		424	传递
302	充分		343	解析		384	项目		425	假定
303	全集		344	计算		385	频率		426	内部
304	公差		345	讨论		386	一直		427	加法
305	公比		346	试问		387	事件		428	劣弧
306	内角		347	象限		388	匀速		429	唯一
307	分成		348	超过		389	图形		430	增加
308	分数		349	过程		390	导数		431	增长
309	切点		350	运算		391	形式		432	完全
310	变化		351	连续		392	排列		433	实根
311	垂线		352	函数		393	数值		434	少于
312	定点		353	曲线		394	数量		435	左边
313	对应		354	速度		395	斜边		436	延长
314	射影		355	任意		396	旋转		437	折线
315	小于		356	部分		397	极限		438	整除
316	小时		357	锐角		398	样本		439	最低
317	平移		358	长度		399	水平		440	焦距

441	甲地		482	求和		523	尺寸		564	吨
442	符合		483	测量		524	属于		565	弧
443	算法		484	画图		525	底部			
444	终止		485	相连		526	弧度			
445	试验		486	等差		527	归纳			
446	调整		487	解答		528	数点			
447	近似		488	负数		529	整理			
448	连结		489	质点		530	方差			
449	阴影		490	路程		531	无理			
450	三等		491	高度		532	正根			
451	两侧		492	三角		533	甲方			
452	两端		493	三边		534	相比			
453	中间		494	上图		535	等待			
454	乘积		495	中线		536	等距			
455	二项		496	乙方		537	简化			
456	仰角		497	互补		538	线性			
457	倾斜		498	位数		539	虚线			
458	减少		499	低于		540	视图			
459	几何		500	余数		541	视角			
460	千克		501	作圆		542	角度			
461	半圆		502	倍数		543	该项			
462	右侧		503	倒数		544	误差			
463	右面		504	倾角		545	起点			
464	四边		505	减法		546	趋向			
465	圆周		506	剩余		547	路径			
466	圆弧		507	割线		548	连线			
467	圆形		508	包含		549	间距			
468	子集		509	包围		550	除数			
469	容量		510	包括		551	除法			
470	底数		511	四周		552	项数			
471	度数		512	回归		553	弓形			
472	恒有		513	图示		554	任取			
473	指数		514	圆点		555	模			
474	数目		515	均值		556	圆			
475	整个		516	定理		557	恒			
476	斜线		517	密度		558	球			
477	无限		518	对数		559	幂			
478	有限		519	对角		560	根			
479	正比		520	对象		561	轭			
480	正面		521	封闭		562	弦			
481	比例		522	小数		563	高			

Appendix B

List of Formula Feature Terms

1	msub19	29	msupq	57	munder	85	times
2	msubt	30	msub20	58	cnx	86	msup0
3	msupm	31	a12	59	s12	87	msubl
4	∑t=0k	32	msub12	60	⋆	88	aman
5	σ22	33	an2	61	ω	89	msubx
6	msupa1	34	fnp	62	g	90	!
7	msupt	35	msup104	63	msub+	91	msupl
8	=	36	msup12	64	k	92	msup
9	∉	37	b	65	<	93	msub10
10	munderover	38	y22	66	msupdot	94	msupx
11	σ	39	fnx	67	msub7	95	msup10
12	m	40	v	68	ins	96	msubΔ
13	msub5	41	x12	69	msup+	97	e
14	q	42	z	70	)	98	msub-
15	msub0.5	43	msub2003	71	msup7	99	γ
16	⋅	44	msub2007	72	[	100	msub18
17	msup5	45	msub2006	73	⊆	101	>
18	x22	46	msub2005	74	/	102	∫01
19	]	47	msub2004	75	mover	103	msub9
20	msub100	48	msupa2	76	lim	104	cnk
21	msube	49	sum	77	log	105	msup-
22	mfrac	50	msubmfrac	78	msupf(4)	106	msup9
23	msubq	51	cnn	79	msuba	107	msubi
24	λ	52	msubj	80	π	108	msubu
25	ln	53	an	81	msub0	109	msupd
26	msup⊥	54	msubsup	82	t	110	msupi
27	int	55	ρ	83	tan	111	cn3
28	.	56	;	84	z2	112	n

113	θ		154	msupk		195	msubbn		236	s
114	f1x		155	(		196	msub13		237	msupf(3)
115	msub2		156	msup24		197	msup1		238	msup3
116	r		157	x3		198	msupbn		239	msubc
117	bm2		158			199	le		240	σ12
118	x2		159	msupx1		200	msupx2		241	dkhl
119	c10000t		160	ξ		201	msubm		242	dkhr
120	msup2		161	Φ		202	msupmfrac		243	cos
121	^		162	msup⋅		203	5amn		244	msupc
122	η		163	∑		204	msupa		245	msupalpha
123	y02		164	ℂ		205	x02		246	alpha
124	msubb		165	l		206	msup0.5		247	msup10000
125	b12		166	msubω		207	msup2005		248	d
126	bar		167	msubg		208	msup2004		249	msubbl
127	msubn		168	msub4		209	msup2006		250	h
128	x04		169	p		210	:		251	msub11
129	msupb		170	msup4		211	uni		252	msub8
130	msupn		171	i		212	f		253	msubk2
131	≥		172	msubd		213	Δ		254	|
132	z1		173	∞		214	φ		255	msupun
133	msupz		174	map		215	cot		256	msup11
134	msub^		175	msubp		216	j		257	msup8
135	dot		176	⊥		217	μ		258	msupan
136	^		177	cnr		218	xn2		259	x
137	msub6		178	max		219	f0x		260	an3
138	sin		179	-		220	x-1		261	fkx
139	c		180	msupp		221	mroot		262	msup3
140	msubπ		181	msup|		222	msup6		263	msubk
141	w		182	↔		223	msubf		264	z3
142	ang		183	msqrt		224	bl		265	msupsin2
143	cnn-2		184	⊂		225	msubr			
144	msup512		185	msub2n		226	cn0			
145	u		186	msupy		227	+			
146	b22		187	x0k		228	⊗			
147	msup/		188	cn1		229	∀			
148	s22		189	a		230	sn2			
149	msubk		190	msub1		231	min			
150	cn2		191	⇒		232	y12			
151	msub		192	msub21		233	msub200			
152	beta		193	msup⇒		234	o			
153	a22		194	y		235	msub3			

Appendix C

List of Publications

The work reported in this thesis has been accepted in the following international conferences.

1. Kai, M. and S.C., Hui and K.Y., Chang. Feature extraction and clustering-based retrieval for mathematical formulas. In *Proceedings of the 2nd International Conference on Software Engineering and Data Mining*, pages 372-377, 2010.

Bibliography

[1] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 41–47, 2003.

[2] K.K. Nam, M.S. Ackerman, and L.A. Adamic. Questions in, knowledge in: A study of naver's question answering community. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, pages 779–788, 2009.

[3] Y. Liu, J. Bian, and E. Agichtein. Predicting information seeker satisfaction in community question answering. In *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 483–490, 2008.

[4] E.M. Voorhees. The TREC question answering track. *Natural Language Engineering*, 7(04):361–378, 2002.

[5] X. Liu, W.B. Croft, and M. Koll. Finding experts in community-based question-answering services. In *Proceedings of the 14th International ACM Conference on Information and Knowledge Management*, pages 315–316, 2005.

[6] E.M. Voorhees and D.M. Tice. Building a question answering test collection. In *Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 200–207, 2000.

[7] R. Srihari and W. Li. Information extraction supported question answering. Technical report, Cymfony Inc., 2000.

[8] E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. In *Proceedings of the 10th International Conference on World Wide Web*, pages 169–178, 2001.

[9] Yahoo Answers! *Available at: http://answers.yahoo.com/.*

[10] Zhidao Baidu. *Available at: http://zhidao.baidu.com/.*

[11] Math Forum at Drexel University. *Available at: http://mathforum.org/.*

[12] Cramster. *Available at: http://www.cramster.com/.*

[13] A. Asperti, F. Guidi, C. Coen, E. Tassi, and S. Zacchiroli. A content based mathematical search engine: Whelp. *Types for Proofs and Programs, Lecture Notes in Computer Science*, 3839:17–32, 2006.

[14] R. Munavalli and R. Miner. Mathfind: a math-aware search engine. In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 735–735, 2006.

[15] Wikipedia Formula Search. *Available at: http://shinh.org/wfs/.*

[16] Wolfram Formula Search. *Available at: http://functions.wolfram.com/formulasearch/.*

[17] A. Youssef. Search of mathematical contents: issues and methods. In *Proceedings of the International Conference on Intelligent and Adaptive Systems and Software Engineering*, pages 100–105, 2005.

[18] J. Hummel, N. Strehmel, J. Selbig, D. Walther, and J. Kopka. Decision tree supported substructure prediction of metabolites from GC-MS profiles. *Metabolomics*, 6(2):322–333, 2010.

[19] Y. Ben-Haim and E. Tom-Tov. A streaming parallel decision tree algorithm. *The Journal of Machine Learning Research*, 11:849–872, 2010.

[20] T. Pedersen. Lexical semantic ambiguity resolution with bigram-based decision trees. *Computational Linguistics and Intelligent Text Processing*, pages 157–168, 2010.

[21] F. Hammann, H. Gutmann, N. Vogt, C. Helma, and J. Drewe. Prediction of adverse drug reactions using decision tree modeling. *Clinical Pharmacology and Therapeutics*, 73:401–406, 2010.

[22] R. Fletcher and G. Zanghirati. Binary separation and training support vector machines. *Acta Numerica*, 19(1):121–158, 2010.

[23] M.H. Tsai, J.D. Chang, S.H. Chiu, and C.H. Lai. Identification of marker genes discriminating the pathological stages in ovarian carcinoma by using support vector machine and systems biology. *Progress in Artificial Life, Lecture Notes in Computer Science*, 4828:381–389, 2010.

[24] A. Sun, M.M. Naing, E.P. Lim, and W. Lam. Using support vector machines for terrorism information extraction. *Intelligence and Security Informatics, Lecture Notes in Computer Science*, 2665:959–959, 2010.

[25] B. Allen, V. Nistor, E. Dutson, G. Carman, C. Lewis, and P. Faloutsos. Support vector machines improve the accuracy of evaluation for the performance of laparoscopic training tasks. *Surgical Endoscopy*, 24(1):170–178, 2010.

[26] Y. Murakami and K. Mizuguchi. Applying the Naive Bayes classifier with kernel density estimation to the prediction of protein-protein interaction sites. *Bioinformatics*, pages 1841–1848, 2010.

[27] T. Calders and S. Verwer. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21:277–292, 2010.

[28] A.M.J. Sarkar, Y.K. Lee, and S. Lee. A smoothed Naive Bayes-based classifier for activity recognition. 27:107–119, 2010.

[29] X. Yang, A. Dinh, and L. Chen. Implementation of a wearerable real-time system for physical activity recognition based on Naive Bayes classifier. In *Proceedings of International Conference on Bioinformatics and Biomedical Technology*, pages 101–105, 2010.

[30] Y. Xiao, M.P. Griffin, D.E. Lake, and J.R. Moorman. Nearest-Neighbor and logistic regression analyses of clinical and heart rate characteristics in the early diagnosis of neonatal sepsis. *Medical Decision Making*, 30:258–266, 2010.

[31] T. Shibata and O. Yamaguchi. Local fisher discriminant component hashing for fast nearest neighbor classification. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 339–349, 2010.

[32] S. Wang and Z. Liu. Infrared face recognition based on histogram and k-Nearest Neighbor classification. *Advances in Neural Networks, Lecture Notes in Computer Science*, 6064:104–111, 2010.

[33] M. Govindarajan and RM Chandrasekaran. Evaluation of k-Nearest Neighbor classifier performance for direct marketing. *Expert Systems with Applications*, 37(1):253–258, 2010.

[34] P. Jurczyk and E. Agichtein. Discovering authorities in question answer communities by using link analysis. In *Proceedings of the 16th ACM International Conference on Information and Knowledge Management*, pages 919–922, 2007.

[35] P. Jurczyk and E. Agichtein. Hits on question answer portals: exploration of link analysis for author ranking. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 845–846, 2007.

[36] J. Zhang, M.S. Ackerman, and L. Adamic. Expertise networks in online communities: Structure and algorithms. In *Proceedings of the 16th International Conference on World Wide Web*, pages 221–230, 2007.

[37] C.S. Campbell, P.P. Maglio, A. Cozzi, and B. Dom. Expertise identification using email communications. In *Proceedings of the 12nd International Conference on Information and Knowledge Management*, pages 528–531, 2003.

[38] K. Balog and M. De Rijke. Determining expert profiles. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2657–2662, 2007.

[39] G.S. Prasad, N.V.S. Reddy, and U.D. Acharya. Knowledge discovery from web usage data: A survey of web usage pre-processing techniques. *Information Processing and Management, Communications in Computer and Information Science*, 70:505–507, 2010.

[40] X. Zhou, S. Chen, B. Liu, R. Zhang, Y. Wang, P. Li, Y. Guo, H. Zhang, Z. Gao, and X. Yan. Development of traditional Chinese medicine clinical data warehouse for medical knowledge discovery and decision support. *Artificial Intelligence in Medicine*, 48:139–152, 2010.

[41] M. Cannataro. Clusters and grids for distributed and parallel knowledge discovery. *High Performance Computing and Networking, Lecture Notes in Computer Science*, 1823:708–716, 2010.

[42] Y. Bi, S. Wu, X. Shen, and J. Guan. Rough analysis for knowledge discovery in a simplified earthquake database. *Integrated Uncertainty Management and Applications, Advances in Soft Computing*, 68:489–500, 2010.

[43] A.J. Cowell, R.S. Jensen, M.L. Gregory, P. Ellis, K. Fligg, L.R. McGrath, K. O'Hara, and E. Bell. Collaborative knowledge discovery and marshalling for intelligence and security applications. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics*, pages 233–238, 2010.

[44] T. Mitchell. Machine learning and data mining. In *Communications of the ACM*, pages 31–36, 1999.

[45] I. Steinwart and A. Christmann. *Support vector machines*. Springer Verlag, 2008.

[46] S.T. Dumais. Latent semantic analysis. *Review of Information Science and Technology*, 38:189–230, 2004.

[47] M. Kohlhase and A. Bundy. *OMDoc–an open markup format for mathematical documents*. Springer, 2002.

[48] LaTeX project. *Available at: http://www.latex-project.org/*.

[49] AsciiMath. *Available at: http://www1.chapman.edu/ jipsen/asciimath.html*.

[50] Mathematical Markup Language. *Available at: http://www.w3.org/TR/MathML2*.

[51] TianYaWenDa. *Available at: http://wenda.tianya.cn/wenda/*.

[52] Google WenDa. *Available at: http://wenda.google.com.hk/wenda/?hl=zh-CN&tab=wH*.

[53] FunAdvice. *Available at: http://www.funadvice.com/*.

[54] WikiAnswers. *Available at: http://wiki.answers.com/*.

[55] Sougou WenDa. *Available at: http://wenda.sogou.com/*.

[56] Askville. *Available at: http://askville.amazon.com/Index.do*.

[57] Answerbag. *Available at: http://www.answerbag.com/*.

[58] LinkedIn Answers. *Available at: http://www.linkedin.com/answers/*.

[59] DaTing. *Available at: http://dating.taobao.com/.*

[60] Ask Bbioo. *Available at: http://www.bbioo.com/.*

[61] MR Hejazi, MS Mirian, K. Neshatian, A. Jalali, and BR Ofoghi. TeLQAS: A telecommunication literature question answering system benefits from a text categorization mechanism. In *Proceedings of the International Conference on Information and Knowedg Engineering*, pages 500–504, 2005.

[62] CSDN. *Available at: http://bbs.csdn.net/.*

[63] R.D. Burke, K.J. Hammond, V. Kulyukin, S.L. Lytinen, N. Tomuro, and S. Schoenberg. Question answering from frequently asked question files: Experiences with the faq finder system. *AI magazine*, page 57, 1997.

[64] S. Lytinen and N. Tomuro. The use of question types to match questions in FAQFinder. In *AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, pages 46–53, 2002.

[65] S. Lytinen, N. Tomuro, and T. Repede. The use of WordNet sense tagging in FAQfinder. In *Proceedings of the AAAI00 Workshop on AI and Web Search*, 2000.

[66] A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–199, 2000.

[67] V. Jijkoun and M. de Rijke. Retrieving answers from frequently asked questions pages on the web. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 76–83, 2005.

[68] A. Jakarta. A high-performance, full-featured text search engine library. *Available at: http://lucene. apache. org*, 2004.

[69] S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. Statistical machine translation for query expansion in answer retrieval. In *Annual Meeting-Association for computational Linguistics*, page 464, 2007.

[70] R. Soricut and E. Brill. Automatic question answering: Beyond the factoid. In *Proceedings of HLT-NAACL*, 2004.

[71] H. Duan, Y. Cao, C.Y. Lin, and Y. Yu. Searching questions by identifying question topic and question focus. *Proceedings of ACL-08: HLT*, pages 156–164, 2008.

[72] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, pages 416–431, 1983.

[73] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *Information Theory, IEEE Transactions on*, pages 2743–2760, 1998.

[74] X. Cao, G. Cong, B. Cui, C.S. Jensen, and C. Zhang. The use of categorization information in language models for question retrieval. In *Proceeding of the 18th ACM conference on Information and knowledge management*, pages 265–274, 2009.

[75] J. Bian, Y. Liu, E. Agichtein, and H. Zha. Finding the right facts in the crowd: factoid question answering over social media. In *Proceeding of the 17th international conference on World Wide Web*, pages 467–476, 2008.

[76] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD workshop on text mining*, page 35, 2000.

[77] F. Diaz. Regularizing ad hoc retrieval scores. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 672–679, 2005.

[78] M.A. Hearst and J.O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84, 1996.

[79] O. Kurland and L. Lee. Corpus structure, language models, and ad hoc information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 194–201, 2004.

[80] X. Liu and W.B. Croft. Cluster-based retrieval using language models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, page 193, 2004.

[81] S.H. Samarasinghe. *Semantic-based retrieval for mathematical knowledge*. Master Thesis. Nanyang Technological University, School of Computer Engineering, 2010.

[82] E. Melis, E. Andres, J. Büdenbender, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. ActiveMath: A generic and adaptive web-based learning environment. *International Journal of Artificial Intelligence in Education*, 12(4):385–407, 2001.

[83] A. Asperti, L. Padovani, C. Coen, and I. Schena. HELM and the semantic math-web. *Theorem Proving in Higher Order Logics, Lecture Notes in Computer Science*, 2152:59–74, 2001.

[84] K. Michael and F. Andreas. MBase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation*, 32(4):365–402, 2001.

[85] A. Franke and M. Kohlhase. System description: MathWeb, an agent-based communication layer for distributed automated theorem proving. *Automated Deduction, Lecture Notes in Computer Science*, 1632:676–676, 1999.

[86] D.W. Lozier. The DLMF project: A new initiative in classical special functions. In *Proceedings of the International Workshop on Special Functions*, pages 207–220, 2000.

[87] MathDi. *Available at: http://mathdi.cesga.es/.*

[88] A.R. Hevner, S.T. March, J. Park, and S. Ram. Design science in information systems research. *Mis Quarterly*, 28(1):75–105, 2004.

[89] Wikipedia. *Available at: http://www.wikipedia.org/.*

[90] PlanetMath. *Available at: http://planetmath.org/.*

[91] S. Wolfram. *The mathematica book*. 1999.

[92] C.D. Manning, P. Raghavan, and H. Schütze. *An introduction to information retrieval*. 2008.

[93] S. Dulucq and H. Touzet. Analysis of tree edit distance algorithms. *Combinatorial Pattern Matching, Lecture Notes in Computer Science*, 2676:83–95, 2003.

[94] I. Rish. An empirical study of the Naive Bayes classifier. In *Proceedings of the IJCAI Workshop on Empirical Methods in Artificial Intelligence*, pages 41–46, 2001.

[95] T. Cover and P. Hart. Nearest Neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[96] V.N. Vapnik. *The nature of statistical learning theory.* Springer Verlag, 2000.

[97] S. Martin, A. Sewani, B. Nelson, K. Chen, and A.D. Joseph. Analyzing behaviorial features for email classification. In *Proceedings of the 2nd Conference on Email and AntiSpam*, 2005.

[98] J. Provost. Naive-Bayes vs. rule-learning in classification of email. *University of Texas at Austin*, 1999.

[99] H.M. Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 977–984, 2006.

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

[101] W.W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems*, 17(2):141–173, 1999.

[102] MerriamWebster. *Available at: http://www.merriam-webster.com/dictionary/blog/.*

[103] X. Qi and B.D. Davison. Web page classification: Features and algorithms. *ACM Computing Surveys*, 41(2):1–31, 2009.

[104] T. Nanno, T. Fujiki, Y. Suzuki, and M. Okumura. Automatically collecting, monitoring, and mining japanese weblogs. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers and Posters*, pages 320–321, 2004.

[105] E. Elgersma and M. De Rijke. Learning to recognize blogs: A preliminary exploration. *EACL Workshop on New Text: Wikis and Blogs and Other Dynamic Text Sources*, pages 24–24, 2006.

[106] R. Mihalcea and H. Liu. A corpus-based approach to finding happiness. In *Proceedings of the AAAI Spring Symposium on Computational Approaches to Weblogs*, pages 139–144, 2006.

[107] P. Chesley, B. Vincent, L. Xu, and R.K. Srihari. Using verbs and adjectives to automatically classify blog sentiment. *In Proceedings of the AAAI Spring Symposium on Computational Approaches to Weblog*, 580(263):27–29, 2006.

[108] G. Mishne. Experiments with mood classification in blog posts. In *Proceedings of the ACM SIGIR Workshop on Stylistic Analysis of Text for Information Access*, 2005.

[109] H. Qu, AL Pietra, and S. Poon. Automated blog classification: Challenges and pitfalls. In *Proceedings of the AAAI Spring Symposium on Computational Approaches to Weblog*, pages 06–07, 2006.

[110] Reuters. *Available at: http://www.reuters.com/.*

[111] PR Newswires. *Available at: http://www.prnewswire.com/.*

[112] C. Chan, A. Sun, and E. Lim. Automated online news classification with personalization. In *Proceedings of the 4th International Conference of Asian Digital Library*, pages 1–10, 2001.

[113] B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory based reasoning. In *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–65, 1992.

[114] D. Billsus and M.J. Pazzani. A hybrid user model for news story classification. In *Proceedings of the 7th International Conference on User Modeling*, pages 99–108, 1999.

[115] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*. Cambridge university press, 1995.

[116] J. Scott. Social network analysis. *Sociology*, page 109, 1988.

[117] L. Li, Y. Shang, and W. Zhang. Improvement of HITS-based algorithms on web documents. In *Proceedings of the 11st International Conference on World Wide Web*, pages 527–535, 2002.

[118] J.C. Miller, G. Rae, F. Schaefer, L.A. Ward, T. LoFaro, and A. Farahat. Modifications of Kleinberg's HITS algorithm using matrix exponentiation and web log records. In *Proceedings of the 24th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 444–445, 2001.

[119] P. Lawrence, B. Sergey, M. Rajeev, and W. Terry. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1999.

[120] A.N. Langville, C.D. Meyer, and P. FernÁndez. Google's PageRank and beyond: The science of search engine rankings. *The Mathematical Intelligencer*, 30(1):68–69, 2008.

[121] B. Dom, I. Eiron, A. Cozzi, and Y. Zhang. Graph-based ranking algorithms for e-mail expertise analysis. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 42–48, 2003.

[122] M. Maybury, R. D'Amore, and D. House. Expert finding for collaborative virtual environments. *Communications of the ACM*, (12):55–56, 2001.

[123] W. Sihn and F. Heeren. Xpertfinder-expert finding within specified subject areas through analysis of e-mail communication. *Proceedings of the Euromedia*, 2001.

[124] A. Mclean, A.M. Vercoustre, and M. Wu. Enterprise PeopleFinder: combining evidence from Web pages and corporate data. In *Proceedings of Australian Document Computing Symposium*, 2003.

[125] C. Macdonald and I. Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 387–396, 2006.

[126] N. Craswell, A. de Vries, and I. Soboroff. Overview of the trec-2005 enterprise track. In *TREC 2005 Conference Notebook*, pages 199–205, 2005.

[127] P. Serdyukov, H. Rode, and D. Hiemstra. Exploiting sequential dependencies for expert finding. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 795–796, 2008.

[128] N. Craswell, D. Hawking, A.M. Vercoustre, and P. Wilkins. P@ noptic expert: Searching for experts not just for documents. In *Ausweb Poster Proceedings, Queensland, Australia*. Citeseer, 2001.

[129] J. Zhang, J. Tang, and J. Li. Expert finding in a social network. *Advances in Databases: Concepts, Systems and Applications, Lecture Notes in Computer Science*, 4443:1066–1069, 2010.

[130] P. Souganidis. Front propagation: Theory and applications. *Viscosity Solutions and Applications, Lecture Notes in Mathematics*, 1660:186–242, 1997.

[131] Y. Zhou, G. Cong, B. Cui, C.S. Jensen, and J. Yao. Routing questions to the right users in online communities. In *Proceedings of the 25th International Conference on Data Engineering*, pages 700–711.

[132] W. Chen, Q. Zeng, L. Wenyin, and T. Hao. A user reputation model for a user-interactive question answering system. *Concurrency and Computation: Practice and Experience*, 19(15):2091–2103, 2007.

[133] J. Zhang, M.S. Ackerman, L. Adamic, and K.K. Nam. QuME: A mechanism to support expertise finding in online help-seeking communities. In *Proceedings of the 20th ACM Symposium on User Interface Software and Technology*, pages 111–114, 2007.

[134] W.C. Kao, D.R. Liu, and S.W. Wang. Expert finding in question-answering websites: A novel hybrid approach. In *Proceedings of the ACM Symposium on Applied Computing*, pages 867–871, 2010.

[135] Yahoo Answers! Tai Wan. *Available at: http://tw.knowledge.yahoo.com/*.

[136] C. Steven. AsciiMathPHP. *Available at: http://www.oldschool.com.sg/index.php/module/Shared/action/Static/tmpl/ASCIIMathPHP*.

[137] K.W. Church and W.A. Gale. Inverse document frequency (IDF): A measure of deviations from Poisson. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 121–130, 1995.

[138] JA Hartigan and MA Wong. A k-means clustering algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28:100–108, 1979.

[139] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, 1983.

[140] T. Kohonen. The self-organizing map. *In Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[141] A. Šilić, M.F. Moens, L. Žmak, and B. Bašić. Comparing document classification schemes using K-means clustering. *Knowledge-based Intelligent Information and Engineering Systems, Lecture Notes in Computer Science*, 5177:615–624, 2010.

[142] M. Mahdavi and H. Abolhassani. Harmony K-means algorithm for document clustering. *Data Mining and Knowledge Discovery*, 18(3):370–391, 2009.

[143] A.K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[144] Y. Zhao, G. Karypis, and U. Fayyad. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168, 2005.

[145] A. El-Hamdouchi and P. Willett. Comparison of hierarchic agglomerative clustering methods for document retrieval. *The Computer Journal*, 32(3):220–227, 1989.

[146] T.W.S. Chow and MKM Rahman. Multilayer SOM with tree-structured data for efficient document retrieval and plagiarism detection. *IEEE Transactions on Neural Networks*, 20(9):1385–1402, 2009.

[147] J. Vesanto and E. Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):586–600, 2000.

[148] F.P. Romero, A. Peralta, A. Soto, J.A. Olivas, and J. Serrano-Guerrero. Fuzzy optimized self-organizing maps and their application to document clustering. *Soft Computing: A Fusion of Foundations, Methodologies and Applications*, 14(8):857–867, 2010.

[149] S. Bandyopadhyay and U. Maulik. An evolutionary technique based on K-means algorithm for optimal clustering in R^N. *Information Sciences*, 146:221–237, 2002.

[150] T. Su and J. Dy. A deterministic method for initializing K-means clustering. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pages 784–786, 2004.

[151] T. Chen, T.H. Tsai, Y.T. Chen, C.C. Lin, R.C. Chen, S.Y. Li, and H.Y. Chen. A combined K-means and hierarchical clustering method for improving the clustering efficiency of microarray. In *Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems*, pages 405–408, 2005.

[152] M. Nanni. Speeding-up hierarchical agglomerative clustering in presence of expensive metrics. *Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science*, 3518:378–387, 2005.

[153] D. Kauchak and S. Dasgupta. An iterative improvement procedure for hierarchical clustering. *Advances in Neural Information Processing Systems*, pages 867–871, 2003.

[154] P. Jackson and I. Moulinier. *Natural language processing for online applications: Text retrieval, extraction and categorization*. John Benjamins Pub Co, 2007.

[155] H. Turtle and W.B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

[156] Gaokao. *Available at: http://en.wikipedia.org/wiki/Gaokao*.

[157] U. Mohideen and A. Roy. Precision measurement of the Casimir force from 0.1 to 0.9 μm. *Physical Review Letters*, 81(21):4549–4552, 1998.

[158] R.B. Yates and B.R. Neto. *Modern information retrieval*. ACM Press, 1999.

[159] H.P. Zhang, H.K. Yu, D.Y. Xiong, and Q. Liu. HHMM-based Chinese lexical analyzer ICTCLAS. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing*, pages 184–187, 2003.

[160] G. Salton and M.J. McGill. *Introduction to modern information retrieval*. McGraw-Hill New York, 1983.

[161] A.S.C. Buckley and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, 1996.

[162] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning*, 1398:137–142, 1998.

[163] T. Kudoh and Y. Matsumoto. Use of support vector learning for chunk identification. In *Proceedings of the 2nd Workshop on Learning Language in Logic*, pages 142–144, 2000.

[164] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

[165] T. Hill and P. Lewicki. *Statistical methods and applications*. StatSoft, Inc., 2005.

[166] N. Cristianini and J. Shawe-Taylor. Support vector machines, 2000.

[167] S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. Technical report, Center for Second Language Research, 2004.

[168] K. Hacioglu and W. Ward. Target word detection and semantic role chunking using support vector machines. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 25–27, 2003.

[169] Schapire R.E. Allwein, E.L. and Y. Singer. Reducing multiclass to binary: A unified approach for margin classifiers. In *Proceedings of the International Conference on Machine Learning*, pages 9–16.

[170] N. Sebe, M.S. Lew, I. Cohen, A. Garg, and T.S. Huang. Emotion recognition using a cauchy Naive Bayes classifier. In *Proceedings of the 16th International Conference on Pattern Recognition*, pages 17–20, 2002.

[171] A.E. Klon, M. Glick, and J.W. Davies. Combination of a Naive Bayes classifier with consensus scoring improves enrichment of high-throughput docking results. *Journal of Medicinal Chemistry*, 47(18):4356–4359, 2004.

[172] T. Denoeux. A k-Nearest Neighbor classification rule based on Dempster-Shafer theory. *IEEE Transactions on Systems, Man and Cybernetics*, 25:804–813, 1995.

[173] X. Yu, K.Q. Pu, and N. Koudas. Monitoring k-Nearest Neighbor queries over moving objects. In *Proceedings of the 21st International Conference on Data Engineering*, pages 631–642, 2005.

[174] V. Van Belle, K. Pelckmans, JAK Suykens, and S. Van Huffel. Support vector machines for survival analysis. In *Proceedings of the 3rd International Conference on Computational Intelligence in Medicine and Healthcare*, 2007.

[175] S. Amari and S. Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12:783–789, 1999.

[176] C.C. Chang and C.J. Lin. LIBSVM- A Library for Support Vector Machines, avaiable at: http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

[177] E.H. Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(2):238–241, 1951.

[178] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1137–1145, 1995.

[179] U.M. Braga-Neto and E.R. Dougherty. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20(3):374–380, 2004.

[180] P. Zhang. Model selection via multifold cross validation. *The Annals of Statistics*, 21(1):299–313, 1993.

[181] P.D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

[182] G.H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[183] O. Alter, P.O. Brown, and D. Botstein. Processing and modeling genome-wide expression data using singular value decomposition. In *Proceedings of the International Society for Optical Engineering*, pages 171–186, 2001.

[184] E.M. Voorhees. The TREC-8 question answering track report. *Nist Special Publication*, 3:77–82, 2000.

[185] E. Voorhees. Overview of the TREC 2001 question answering track. *Nist Special Publication*, 4:42–51, 2002.

[186] E. Voorhees and DM Tice. Overview of the TREC-9 question answering track. *Nist Special Publication*, 3:71–80, 2001.

[187] A. Kuhn, S. Ducasse, and T. Gīrba. Semantic clustering: Identifying topics in source code. *Information and Software Technology*, 49(3):230–243, 2007.

[188] K. Ma, S.C. Hui, and K.Y. Chang. Feature extraction and clustering-based retrieval for mathematical formulas. In *Proceedings of the 2nd International Conference on Software Engineering and Data Mining*, pages 372–377, 2010.

[189] M. Adriani, Rijsbergen, and van C. J. Term similarity-based query expansion for cross-language information retrieval. In *Proceedings of the 3rd European Conference on Research and Advanced Technology for Digital Libraries*, pages 311–322, 1999.

[190] D. Hiemstra and F. De Jong. Disambiguation strategies for cross-language information retrieval. *Research and Advanced Technology for Digital Libraries, Lecture Notes in Computer Science*, 1696:852–852, 2010.

[191] B. Roth and D. Klakow. Combining Wikipedia-based concept models for cross-language retrieval. *Advances in Multidisciplinary Retrieval, Lecture Notes in Computer Science*, 6107:47–59, 2010.

[192] D. Mandal, S. Dandapat, M. Gupta, P. Banerjee, and S. Sarkar. Bengali and Hindi to English cross-language text retrieval under limited resources. In *Proceedings of Working Notes for the CLEF Workshop*, 2007.

[193] P.A. Chew, B.W. Bader, T.G. Kolda, and A. Abdelali. Cross-language information retrieval using PARAFAC2. In *Proceedings of the 13th International ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 143–152, 2007.

[194] A-Level Mathematics. *Available at: http://en.wikipedia.org/wiki/Advanced_level_mathematics*.

[195] TouchGraph. *Available at: http://www.touchgraph.com/TGGoogleBrowser.html*.

[196] RGraph. *Available at: http://drupal.org/project/rgraph*.

[197] M. Wattenberg and F.B. Viégas. The Word Tree, an interactive visual concordance. *IEEE Transactions on Visualization and Computer Graphics*, 14:1221–1228, 2008.

[198] EntityCube. *Available at: http://entitycube.research.microsoft.com/*.

[199] WonderWheel. *Available at: http://www.googlewonderwheel.com/*.

[200] TimeLine. *Available at: http://code.google.com/apis/visualization/documentation/gallery/annotatedtimeline.html*.

[201] ImageSwirl. *Available at: http://image-swirl.googlelabs.com/*.

[202] R.C. Gershon. Computer adaptive testing. *Journal of Applied Measurement*, 6(1):109–127, 2005.

[203] H. Wainer, NJ Dorans, D. Eignor, R. Flaugher, BF Green, RJ Mislevy, and L. Steinberg. Computerized adaptive testing: A primer. *Quality of Life Research*, 10(8):733–734, 2001.

[204] M.E. Lunz, B.A. Bergstrom, and R.C. Gershon. Computer adaptive testing. *International Journal of Educational Research*, 21(6):623–634, 1994.

[205] W.J. Van Der Linden and C.A.W. Glas. *Computerized adaptive testing: Theory and practice*. Springer Netherlands, 2000.

[206] E. Latu and E. Chapman. Computerised adaptive testing. *British Journal of Educational Technology*, 33(5):619–622, 2002.