Semantic-based retrieval for mathematical knowledge

Sidath Harshanath Samarasinghe

2010

Sidath Harshanath Samarasinghe. (2010). Semantic-based retrieval for mathematical knowledge. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/47471>

<https://doi.org/10.32657/10356/47471>

Nanyang Technological University

Downloaded on 26 Apr 2023 23:11:06 SGT

SEMANTIC-BASED RETRIEVAL FOR MATHEMATICAL KNOWLEDGE

SIDATH HARSHANATH SAMARASINGHE

School of Computer Engineering

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

2010

ACKNOWLEDGEMENTS

The author would like to express his special thanks to Dr. Hui Siu Cheung, the author's supervisor, for his constant encouragement, very helpful guidance and thoughtful criticism given throughout this project. Without his strong support, patience and enlightenment, the completion of this thesis would have not been possible. He has extended his help beyond the call of duty, and the author is truly grateful for all of the help given by him.

The author also wants to thank Mr. Huang Tingfang for providing his initial thoughts on this project's development, and Mr. Doan Minh Tuan and Mr. Ma Kai for their help in the data creation tasks in this project.

Finally, the author likes to thank the technical staff at the Center for Advanced Information Systems (CAIS) for their support and providing the required laboratory facilities for carrying out this project.

And many thanks to the author's family for their support and patience, and all other people who have helped in this project one way or the other.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS..... i

TABLE OF CONTENTS..... ii

LIST OF FIGURES..... v

LIST OF TABLES..... viii

ABSTRACT..... ix

CHAPTER 1 INTRODUCTION..... 1

1.1 MOTIVATION..... 1

1.2 FORMULA SEARCH..... 3

1.3 MATHEMATICAL DOCUMENT RETRIEVAL..... 4

1.4 OBJECTIVES..... 4

1.5 ORGANIZATION OF THE THESIS..... 6

CHAPTER 2 DOCUMENT RETRIEVAL..... 7

2.1 INFORMATION RETRIEVAL..... 7

2.2 VECTOR SPACE MODEL..... 9

2.2.1 Document Indexing..... 9

2.2.2 Term Weighting..... 10

2.2.3 Document Retrieval and Ranking..... 11

2.3 CLUSTERING TECHNIQUES..... 11

2.3.1 Hierarchical Clustering Algorithms..... 12

2.3.2 Partitional Clustering Algorithms..... 13

2.3.3 Neural Network-based Clustering Techniques..... 14

2.3.4 Hybrid Clustering Techniques..... 15

2.4 SUMMARY..... 16

CHAPTER 3 MATHEMATICAL DOCUMENTS..... 18

3.1 NOTION OF MATHEMATICAL DOCUMENT..... 18

Table of Contents

3.2	FEATURES OF MATHEMATICAL DOCUMENTS.....	19
3.2.1	Plain Text Descriptions.....	20
3.2.2	Diagrams.....	20
3.2.3	Mathematical Formulas.....	21
3.3	SUMMARY.....	26
 CHAPTER 4 FORMULA SEARCH.....		27
4.1	FORMULA SEARCH CONSIDERATIONS.....	27
4.2	FORMULA SEARCH SYSTEMS.....	29
4.3	TREE EDIT DISTANCE.....	33
4.3.1	Zhang-Shasha Algorithm.....	35
4.3.2	Discussion.....	36
4.4	PROPOSED APPROACH.....	37
4.4.1	Formula Preprocessing and Transformation.....	39
4.4.2	Formula Clustering.....	43
4.4.3	The Formula Knowledge Base.....	51
4.4.4	Query Preprocessing and Transformation.....	51
4.4.5	Cluster Identification.....	52
4.4.6	Expression Tree Construction.....	53
4.4.7	Formula Matching and Ranking.....	60
4.5	PERFORMANCE EVALUATION.....	70
4.5.1	Experimental Setup.....	71
4.5.2	Training Efficiency.....	74
4.5.3	Retrieval Performance.....	79
4.6	SUMMARY.....	88
 CHAPTER 5 MATHEMATICAL DOCUMENT RETRIEVAL.....		90
5.1	PROPOSED APPROACH.....	90
5.1.1	Feature Selection.....	92
5.1.2	Preprocessing.....	94
5.1.3	Transformation.....	98
5.1.4	Document Clustering.....	98
5.1.5	Document Knowledge Base.....	99
5.1.6	Query Feature Selection and Preprocessing.....	100
5.1.7	Query Transformation.....	100

Table of Contents

5.1.8	Cluster Identification.....	100
5.1.9	Document Ranking.....	101
5.2	PERFORMANCE EVALUATION.....	101
5.2.1	Training Efficiency.....	103
5.2.2	Retrieval Performance.....	108
5.3	SUMMARY.....	111
CHAPTER 6 CONCLUSION AND FUTURE WORK.....		113
6.1	SUMMARY.....	113
6.2	CONTRIBUTIONS.....	117
6.3	FUTURE WORK.....	117
6.3.1	Incorporating Features from Problem Solutions into the Clustering Process.....	118
6.3.2	Visualization of Search Results.....	118
6.3.3	Incorporating an Improved Formula Search Mechanism for Mathematical Document Retrieval.....	119
6.3.4	Conducting a Usability Study.....	119
REFERENCES.....		120
APPENDIX A LIST OF MATHEMATICAL REGULAR EXPRESSIONS.....		132
APPENDIX B LIST OF COMPOSITE MATHML ELEMENTS.....		155
APPENDIX C FORMULA DATA SET FOR MATCHING ACCURACY EVALUATION.....		156
APPENDIX D 60 TEST QUERIES FOR PERFORMANCE EVALUATION OF FORMULA SEARCH.....		163
APPENDIX E LIST OF INDICATIVE KEYWORDS.....		168
APPENDIX F 60 TEST QUERIES FOR MATHEMATICAL DOCUMENT RETRIEVAL EVALUATION.....		172

LIST OF FIGURES

FIGURE 1-1: TWO MAJOR FORMS OF MATHEMATICAL PROBLEMS.....	2
FIGURE 1-2: TWO MAIN CLUSTERING TASKS.....	5
FIGURE 2-1: DOCUMENT-TERM MATRIX IN VSM.	10
FIGURE 2-2: DOCUMENTS A, B, AND C REPRESENTED AS VECTORS IN THE VECTOR SPACE.....	10
FIGURE 3-1: A MATHEMATICAL DOCUMENT.....	19
FIGURE 3-2: EXAMPLE OF A DIAGRAM.	20
FIGURE 3-3: STRUCTURAL NATURE OF MATHEMATICAL FORMULAS.....	21
FIGURE 4-1: NODE EDIT OPERATIONS ON TREES.....	34
FIGURE 4-2: KRS AND LMLDS OF A TREE.....	36
FIGURE 4-3: PROPOSED MATHEMATICAL FORMULA RETRIEVAL APPROACH.....	38
FIGURE 4-4: FORMULA PREPROCESSING AND TRANSFORMATION STEP...	40
FIGURE 4-5: EXAMPLE OF FORMULA PREPROCESSING AND TRANSFORMATION.....	43
FIGURE 4-6: THE WORK FLOW OF SOM+K-MEANS HYBRID CLUSTERING TECHNIQUE.....	50
FIGURE 4-7: CLUSTER IDENTIFICATION.....	53
FIGURE 4-8: EXPRESSION TREE REPRESENTATION OF A MATHEMATICAL FORMULA.....	55
FIGURE 4-9: NORMALIZATION OF MATHML REPRESENTATIONS INTO ROOT FORM.....	56
FIGURE 4-10: SET OF MATHML ELEMENTS FOR EXPRESSION TREE CONSTRUCTION.....	57
FIGURE 4-11: PSEUDO-CODE FOR EXPRESSION TREE CONSTRUCTION.....	58
FIGURE 4-12: EXPRESSION TREE CONSTRUCTION.....	59
FIGURE 4-13: FORMULA MATCHING AND RANKING.....	61
FIGURE 4-14: TWO EXPRESSION TREES AND THEIR LMLDS AND KRS.....	62
FIGURE 4-15: TREE EDIT DISTANCE COMPUTATION BETWEEN TWO EXPRESSION TREES.....	63
FIGURE 4-16: TWO SIMILAR FORMULAS HAVING DISTINCTIVE EXPRESSION TREES.....	66

List of Figures

FIGURE 4-17: TWO CANONICALLY EQUIVALENT MATHEMATICAL FORMULAS HAVING DIFFERENT MATHML REPRESENTATIONS.....	68
FIGURE 4-18: SCREENSHOT OF THE FORMULA SEARCH ENGINE PROTOTYPE.....	71
FIGURE 4-19: TRAINING EFFICIENCY OF AHC.....	74
FIGURE 4-20: TRAINING EFFICIENCY OF K-MEANS.....	75
FIGURE 4-21: TRAINING EFFICIENCY FOR DIFFERENT NUMBERS OF TRAINING ITERATIONS CONFIGURED ON SOM.....	76
FIGURE 4-22: TRAINING EFFICIENCY FOR DIFFERENT NUMBERS OF NEURONS CONFIGURED ON SOM.....	77
FIGURE 4-23: TRAINING EFFICIENCY OF SOM+K-MEANS FOR DIFFERENT NUMBERS OF TRAINING ITERATIONS CONFIGURED ON SOM.....	78
FIGURE 4-24: TRAINING EFFICIENCY OF SOM+K-MEANS FOR DIFFERENT NUMBERS OF NEURONS CONFIGURED ON SOM	78
FIGURE 4-25: TESTING RETRIEVAL PERFORMANCE FOR AHC AND K-MEANS.....	85
FIGURE 4-26: TESTING RETRIEVAL PERFORMANCE FOR SOM AND SOM+K-MEANS ACCORDING TO THE NUMBER OF TRAINING ITERATIONS CONFIGURED ON SOM.....	86
FIGURE 4-27: TESTING RETRIEVAL PERFORMANCE FOR SOM AND SOM+K-MEANS ACCORDING TO THE NUMBER OF NEURONS CONFIGURED ON SOM.....	86
FIGURE 5-1: PROPOSED MATHEMATICAL DOCUMENT RETRIEVAL APPROACH.....	91
FIGURE 5-2: EXAMPLE OF QUESTION DIAGRAM DESCRIBED IN THE TEXT DESCRIPTION.....	93
FIGURE 5-3: EXAMPLE OF FEATURE SELECTION STEP.....	93
FIGURE 5-4: SNAPSHOT OF THE INDICATIVE KEYWORD LIST.....	96
FIGURE 5-5: PREPROCESSING STEP EXAMPLE.....	97
FIGURE 5-6: TRAINING EFFICIENCY OF AHC.....	103
FIGURE 5-7: TRAINING EFFICIENCY OF K-MEANS.....	104
FIGURE 5-8: TRAINING EFFICIENCY FOR DIFFERENT NUMBERS OF TRAINING ITERATIONS CONFIGURED ON SOM.....	105
FIGURE 5-9: TRAINING EFFICIENCY FOR DIFFERENT NUMBERS OF NEURONS CONFIGURED ON SOM.....	105

List of Figures

FIGURE 5-10: TRAINING EFFICIENCY OF SOM+K-MEANS FOR DIFFERENT NUMBERS OF TRAINING ITERATIONS CONFIGURED ON SOM.....	106
FIGURE 5-11: TRAINING EFFICIENCY OF SOM+K-MEANS FOR DIFFERENT NUMBERS OF NEURONS CONFIGURED ON SOM	107
FIGURE 5-12: TESTING RETRIEVAL PERFORMANCE FOR AHC AND K-MEANS.....	109
FIGURE 5-13: TESTING RETRIEVAL PERFORMANCE FOR SOM AND SOM+K-MEANS ACCORDING TO THE NUMBER OF TRAINING ITERATIONS CONFIGURED ON SOM.....	110
FIGURE 5-14: TESTING RETRIEVAL PERFORMANCE FOR SOM AND SOM+K-MEANS ACCORDING TO THE NUMBER OF NEURONS CONFIGURED ON SOM.....	110

LIST OF TABLES

TABLE 3-1: REPRESENTATION OF A MATHEMATICAL FORMULA IN
THREE ENCODING FORMATS..... 25

TABLE 4-1: SUMMARY OF PROMINENT MATHEMATICAL FORMULA
SEARCH SYSTEMS..... 32

TABLE 4-2: EXAMPLES OF MATHEMATICAL REGULAR EXPRESSIONS.... 41

TABLE 4-3: INFORMATION STORED IN THE FORMULA
KNOWLEDGE BASE..... 51

TABLE 4-4: CANONICAL EQUIVALENCES IN FORMULAS..... 69

TABLE 4-5: COMPOSITION OF THE INTEGRATION FORMULA DATA SET.. 73

TABLE 4-6: EXPERIMENTAL RESULTS FOR THE EVALUATION OF wf 83

TABLE 4-7: MATCHING ACCURACY RESULTS..... 83

TABLE 4-8: SUMMARY OF BEST TRAINING RETRIEVAL PERFORMANCE
FOR AHC, K-MEANS, SOM, AND SOM+K-MEANS
CLUSTERING TECHNIQUES..... 84

TABLE 4-9: SUMMARY OF BEST TESTING RETRIEVAL PERFORMANCE
FOR AHC, K-MEANS, SOM, AND SOM+K-MEANS
CLUSTERING TECHNIQUES..... 87

TABLE 5-1: INFORMATION STORED IN THE DOCUMENT
KNOWLEDGE BASE..... 99

TABLE 5-2: COMPOSITION OF THE MATHEMATICAL DOCUMENT
DATA SET..... 102

TABLE 5-3: SUMMARY OF BEST TRAINING RETRIEVAL PERFORMANCE
FOR AHC, K-MEANS, SOM, AND SOM+K-MEANS
CLUSTERING TECHNIQUES..... 108

TABLE 5-4: SUMMARY OF BEST TESTING RETRIEVAL PERFORMANCE
FOR AHC, K-MEANS, SOM, AND SOM+K-MEANS
CLUSTERING TECHNIQUES..... 111

ABSTRACT

Mathematics is an important subject learned by students studying in schools and universities. Students may acquire mathematical knowledge in different ways. One of the most effective ways is through solving mathematical problems. But, solving mathematical problems is a challenging task for many students. Therefore, Computer Assisted Learning (CAL) tools such as Computer Algebra Systems (CASs) have been introduced in order to help students to solve their mathematical problems. However, the currently available CASs are unable to provide the solutions to most mathematical problems, which usually contain a problem description in text, together with mathematical formulas and/or diagrams.

This research intends to overcome the issues faced in using CASs and proposes an efficient and effective approach that can be used by students to help in solving their mathematical problems. The proposed approach is basically an information retrieval approach that works as follows. The user issues a mathematical problem in the form of a query to a mathematical document search engine. Then, the search engine retrieves a set of similar mathematical problems together with their associated solutions. In mathematics, it is a known fact that the mathematical problems that are similar also have similar solutions. Therefore, the retrieved similar problems and their solutions are most likely useful to the user for solving his original problem.

This research aims to tackle two major issues related to retrieving mathematical content. One is formula search, which searches similar mathematical formulas for a given formula. The other is document search, which searches similar mathematical documents for a given query. A mathematical document is considered a composite object consisting of a mathematical problem and its corresponding solution.

Abstract

In addition, this research studies in depth the use of clustering techniques for both formula and document search. Four different types of clustering techniques, namely, Agglomerative Hierarchical Clustering (AHC), K-means, Kohonen's Self-Organizing Maps (SOM), and a hybrid clustering technique that integrates the SOM neural network with the K-means algorithm (SOM+K-means) are employed for the formula search engine and the mathematical document retrieval engine. Their performance is evaluated for both formula and document search.

At the end of this thesis, possible areas of future work are briefly discussed. They include incorporating semantic features from the problem solutions into the clustering process, visualization of search results, and conducting a usability study.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Students in schools and Universities generally encounter mathematical problems in their studies; in day-to-day work (e.g., exercises), during revision, and in examinations. Solving mathematical problems is a challenging task faced by many students. When a student finds it difficult to solve a particular mathematical problem with his existing knowledge, he will usually refer to the text books for related examples, or he would consult his peers or someone with expert knowledge in the area. However, such an approach is quite inefficient and sometimes students come across many difficulties in locating the sources that can help them in solving their mathematical problems. Therefore, an efficient and effective approach that can help students to solve their mathematical problems would be of great practical and pedagogical value.

Currently, there exist a number of computer algebra systems (CASs) such as Maple [66] and Mathematica [67], which can provide step-by-step solutions to certain types of mathematical problems. However, these systems require the mathematics to be entered in symbolic notation (i.e., mathematical formula or mathematical expression) [114]. They are unable to provide the solutions to the vast majority of mathematical problems that comprise a problem description, mathematical formulas, and diagrams. Therefore, CASs are not very practical for solving mathematical problems, except for the expert CAS users.

Alternatively, search engines can also be used as a means for problem solving [5, 63]. That is because when we search for information, we engage in a problem solving process. However, the general search engines available nowadays such as Google [29] and Yahoo [128] are unable to provide hints or help for mathematical problem solving. In addition, the few available mathematical

document retrieval systems, such as Math Search Project [131] and Zentralblatt Math [51], are not developed with the objective of mathematical problem solving.

This research aims to investigate an efficient and effective solution for mathematical problem solving. In this research, we propose a document retrieval approach for solving mathematical problems. The main hypothesis behind our approach is that *similar mathematical problems are solved using similar techniques*. That means, if we can find a very similar problem to a user's query problem, the solution to the similar problem can most likely be used to provide hints or solutions on solving the user's query problem. Based on a user's query problem, a similar set of mathematical problems together with their associated solutions are retrieved in order to provide hints or solutions on solving the user's query problem. In this research, a mathematical document is defined as a mathematical problem and its corresponding solution.

Figure 1-1: Two major forms of mathematical problems.

In general, a mathematical problem can be specified in one of two main forms as shown in Figure 1-1 [100, 111]. The first form is the *formula problem*, which is a problem consisting primarily a mathematical formula. To be precise, the question is mainly about solving a mathematical formula. The second form is the *document problem*, which is a mathematical problem that contains a problem description, and (optionally) mathematical formulas and diagrams. Corresponding to the two different forms of mathematical problems, two different search tasks can be defined: formula search and mathematical document retrieval.

1.2 Formula Search

Mathematical formulas (sometimes called *mathematical expressions* or *mathematical notation*) are perhaps the key feature in mathematical documents. A formula is a symbolic way of expressing a certain mathematical concept or idea. Mathematical formulas probably constitute a significant proportion of semantic information in mathematical documents. That is because a formula is typically composed of several small mathematical elements (e.g., mathematical symbols, operators, etc.), and each mathematical element carries some semantic meaning, which collectively leads to a lot of semantics. For example, consider the formula: $\int \sqrt{x^2 + 2x} \, dx$. It consists of mathematical elements such as the Integral sign ($\int$), variable (x), power (2) (i.e., in x^2), constant (2), addition operator ($+$), and the square root symbol ($\sqrt{\quad}$). As such, there is a lot of semantics represented by this formula although its visual appearance seems quite compact. In addition, formulas are also quite frequent in mathematical documents, which also mean they add a great deal of semantics into mathematical documents.

Many full-text retrieval systems with quite mature technology are available nowadays. However, the techniques adopted in those systems are not effective for searching mathematical formulas. There are several reasons for this [74, 76, 92]. One of the main reasons is because of the structural nature of mathematical formulas. That is, the structure of a mathematical formula provides important information, which becomes void (or changed) if the structure is lost (or changed).

For example, the two formulas: x^2 and $2x$ consists of the same set of characters $\{2, x\}$, but the meaning of x^2 is completely different from that of $2x$. The currently available full-text retrieval systems cannot handle such structured mathematical text (in fact, they would probably treat both x^2 and $2x$ in a similar way). In addition, the querying methods of existing full-text retrieval systems do not support the entry of structured mathematical text or any symbolic mathematical elements, such as the Integral sign. For such reasons, we need special techniques for searching mathematical formulas.

1.3 Mathematical Document Retrieval

The notion of *mathematical document* may be given different interpretations depending on the context of use. In this research, we refer to a mathematical document as a composite object that consists of a mathematical problem and its corresponding solution. In general, mathematical documents are formed by text descriptions, mathematical formulas, and diagrams. Therefore, to search for mathematical documents effectively, techniques that can handle all of these different types of features are essential.

1.4 Objectives

The main objective of this research is to propose a document retrieval approach for mathematical problem solving. Our idea is that, based on a user's query problem, similar mathematical problems with their associated solutions are retrieved, which can then be used to provide hints or solutions on solving the user's query problem.

In this project, we aim to tackle two main issues related to mathematical document retrieval. One is the formula search, which is the retrieval of similar mathematical formulas to a given formula. The other is the mathematical document retrieval, which is the retrieval of mathematical documents that comprises one or

more of the following: problem description, mathematical formulas, and graphical figures (i.e., diagrams).

This research particularly focuses on applying clustering techniques for the intelligent retrieval of mathematical knowledge [88]. Since we have two different retrieval tasks as mentioned above, we also have two different clustering tasks, as shown in Figure 1-2. One is formula clustering, where clustering techniques are applied to a collection of mathematical formulas stored in a formula database. The other is document clustering, where clustering techniques are applied to a collection of mathematical documents stored in a mathematical document database.

Figure 1-2: Two main clustering tasks.

- *Formula Clustering.* In Formula Clustering, clustering techniques are applied for semantic clustering of similar mathematical formulas. Four different types of clustering techniques are investigated, namely, Agglomerative Hierarchical Clustering (AHC) [77], K-means [134], Kohonen's Self-Organizing Maps (SOM) [54], and a hybrid clustering technique that integrates the SOM with the K-means algorithm (SOM+K-means) [59, 60].
- *Document Clustering.* In Document Clustering, clustering techniques are applied in order to group similar mathematical documents. This is based on the underlying assumption that mathematical problems with similar features tend to be solved using similar techniques. AHC, K-means, SOM, and SOM+K-means clustering techniques are explored for document clustering as well.

1.5 Organization of the Thesis

This chapter describes the motivation behind this research work. It briefly discusses the challenges involved in searching mathematical documents. The two key issues in retrieving mathematical documents are introduced: mathematical formula search and mathematical document retrieval. The objectives of this research are then discussed. The rest of this thesis is organized as follows.

In Chapter 2, we discuss the related background on document retrieval. It discusses information retrieval (IR), various IR models, and the popular Vector Space Model (VSM) for IR. In addition, an overview on clustering techniques is also given, where we discuss the different clustering techniques applied in this project.

Chapter 3 discusses the structure and the features of mathematical documents.

Formula search is discussed in Chapter 4. It includes a detailed discussion on various considerations related to formula retrieval, related existing systems, and techniques applied in this research. Here we propose a new approach for mathematical formula retrieval, which is based on semantic clustering and formula matching using the *tree edit distance* technique. Four different types of clustering techniques, AHC, K-means, SOM, and SOM+K-means, are explored for formula clustering.

Chapter 5 presents our proposed approach for mathematical document retrieval. The clustering techniques explored for formula search (AHC, K-means, SOM, and SOM+K-means) are explored for mathematical document clustering as well.

Finally, Chapter 6 gives the concluding remarks of this research work and indicates the possible areas for future work.

CHAPTER 2

DOCUMENT RETRIEVAL

Since this project aims to investigate information retrieval approaches for mathematical problem solving, this chapter first reviews the various techniques and concepts related to document retrieval as applied to this research work. The chapter begins with a discussion on the notion of Information Retrieval (IR), which is the primary concept behind this research work. Then, the popular Vector Space Model (VSM) of information retrieval is discussed in detail, which is used by many document clustering methods. As this research particularly focuses on using clustering techniques for document retrieval, a detailed discussion on the related clustering techniques is given next. Finally, a brief discussion on the ways in which the concepts discussed in this chapter relate to this research work is given.

2.1 Information Retrieval

Information Retrieval (IR) [65] refers to finding information from large collections of documents, which are typically unstructured in nature, in order to satisfy an information need of a user. The terms “information” and “document” are typically used interchangeably since information is most commonly accessible in discrete units, which are referred to as *documents*. IR deals with the representation, storage, organization, and access to documents [129] that are usually stored in a database or some form of information repository. A typical IR task begins with the information need of a user, which he/she realizes in the form of a query. The user’s query can then be input to a search engine or information retrieval system (IRS) (sometimes called *search engine*) in order to find relevant documents that can meet the user’s information need [42].

A concept that is central to the measurement of “goodness” of an IR outcome is the notion of *relevance* [71]. In general, the goal of an IR task is to retrieve the relevant documents from the document collection, while retrieving none or as few numbers of non-relevant documents as possible. Two most

commonly used relevance measures are subjective relevance and utility-based relevance [71]. Subjective relevance determines a document as relevant to a query if the pre-assigned subject of the document matches with the subject of the query. Conversely, utility-based relevance gauges a document as relevant to a query if the document is useful to the user who issued the query.

Traditionally IR has dealt a lot with natural language text, which is not always well-structured, and can often be semantically ambiguous [129]. As such, an IRS typically needs to ‘interpret’ the content of the documents in order to process them to satisfy the user’s information need. To ‘interpret’ the content of the documents, the IRS needs to extract the syntactic and semantic information from the document’s content, and then use that information to match with the syntactic and semantic information of the user’s query.

Nowadays, IR is applied to a vast number of fields, for instance geography, medicine, library science, and news-and-media, to retrieve different kinds of information. For example, [24] discusses medical IR, while [57] investigates geographic IR. The recent past has seen an emerging trend in applying IR for the retrieval of mathematical knowledge from collections of mathematical documents or mathematical document databases [4, 8, 30]. Since mathematical knowledge is typically composed of structured mathematical text and mathematical operators [76], mathematical knowledge retrieval is considered a specialized search task [132] that requires the incorporation of domain knowledge [135] and special techniques in order to index, retrieve, and present mathematical knowledge for the users.

The evolution of IR is signified by the introduction of various IR models. Basically there are three classical IR models [129] that provide three different frameworks for the representation of documents and queries, namely, Boolean model, Vector Space Model, and the probabilistic retrieval model. The Boolean model is the simplest among the three models. It is based on the Boolean logic and set theory, and conceives the documents and the user’s query as a set of index terms. The retrieval is based on whether or not the index terms found in the documents are also found in the query. In the Vector Space Model (VSM), the

documents and the user's query are expressed using term frequency vectors in an n -dimensional space with the set of all index terms forming the vector space. It is also referred to as an 'algebraic' model because it uses algebra as the basis of representing elements in the model. The third model is the probabilistic retrieval model, which uses probability theory as the core concept for the representation of documents and queries, and uses probabilistic measures in calculating similarity values and relevance measures in the model.

2.2 Vector Space Model

Among the three classical IR models, perhaps the most popular model is the Vector Space Model (VSM), which was proposed by Salton [87]. In VSM, documents are represented as feature vectors. Many IR tasks (typically those that involve text such as text categorization and text clustering) use VSM as the basic framework for the representation of documents [80, 91]. The VSM can be described basically in terms of three main phases. In the first phase, index terms are extracted from documents and each document is then represented using a document vector in an n -dimensional space of terms. In the second phase, index terms are weighted using a weighting function in order to improve the effectiveness of the retrieval. During the third phase, relevant documents are retrieved in response to a user's query, which are eventually ranked using a chosen similarity measure. Next we will describe these three phases in more details.

2.2.1 Document Indexing

In the Document Indexing phase, each document in the system is represented as an n -dimensional vector, where each dimension corresponds to an index term found in the system's document corpus. The presence of an index term in a document results in a non-zero value for the particular dimension representing the frequency of occurrence of the index term in the document. On the other hand, an index term that does not occur in a document is given a value of zero for its respective dimension. This is illustrated with an example in Figure 2-1 [18].

Figure 2-1: Document-term matrix in VSM.

In the example shown in Figure 2-1, there are three documents in the system’s document collection, with document id’s A, B, and C. Each document discusses a specific subject, and hence contains different terms (with different term frequencies). The above information can be expressed as a series of vectors in a pictorial form as shown in Figure 2-2.

Figure 2-2: Documents A, B, and C represented as vectors in the vector space.

2.2.2 Term Weighting

Once the documents are represented as feature vectors, a weighting function is used to assign weights to the index terms. One of the most commonly used weighting scheme together with VSM is the TFIDF (Term Frequency Inverse Document Frequency) [87] weighting scheme.

2.2.3 Document Retrieval and Ranking

In the third and the final phase, the user issues a query. The user query is converted to a vector representation using a similar procedure applied in the Document Indexing phase. Afterwards, a similarity measure is used to find documents that are similar to the user's query, and rank them according to their similarity to the user's query. Examples of similarity measures that can be used for ranking include the Euclidean distance and the cosine similarity [64, 122].

2.3 Clustering Techniques

Clustering is an unsupervised learning task [61, 81, 105] in which the objective is to find groups of similar objects that share a number of interesting properties. A cluster is a collection of objects (or records) that are "similar" to one another and "dissimilar" to objects in other clusters. The idea of clustering has been initially envisaged as a means of clustering documents into groups of related documents in the field of information retrieval, which was pioneered by van Rijsbergen who formulated the *Cluster Hypothesis* [118]: "Closely associated documents tend to be relevant to the same requests". The term "documents" mentioned here consist of a set of words and the words with discriminating features form the key to clustering [45]. The result is that it produces clusters of documents by discriminating relevant documents from the irrelevant documents.

A clustering task is achieved through a clustering algorithm (or clustering technique), which seeks to segment the entire data set into relatively homogenous subgroups (i.e., clusters) such that the similarity of the objects within a cluster is maximized and the similarity of the objects outside the cluster is minimized. In general, all clustering techniques can be categorized into two basic categories: hierarchical and non-hierarchical (also called partitional) [9, 43]. In the recent past, a few other flavours of clustering techniques have also emerged, such as neural network-based clustering techniques and hybrid clustering techniques that integrate different clustering algorithms.

In the following sections, we will briefly discuss the abovementioned types of clustering algorithms, hierarchical, partitional, neural network-based, and hybrid approaches.

2.3.1 Hierarchical Clustering Algorithms

Hierarchical clustering algorithms are relatively classic compared to the other types of clustering algorithms, but they are still in widespread use. In hierarchical clustering algorithms, a tree-like structure (called a *dendrogram*) is created by combining existing clusters (agglomerative methods) or through recursive partitioning (divisive methods) [65]. Agglomerative methods operate in a bottom-up fashion by starting with each data point (or object) as an individual cluster, and at each subsequent step, combines the two closest pair of clusters to form a new cluster. In contrast, divisive methods operate in a complementary top-down approach, starting with a single cluster containing all the objects, and at each step, splits a cluster until only clusters with single objects remain.

Out of the two major types of hierarchical clustering techniques, agglomerative methods are used more often than divisive methods due to the difficulty in searching for partitions in the divisive methods [61]. Agglomerative methods require a definition for the distance between clusters, known as *cluster proximity*. Based on the method used to calculate the cluster proximity, there are three basic variations of agglomerative hierarchical clustering techniques, namely, single link (minimum distance), complete link (maximum distance), and the group average method [48, 102]. There is also another technique called Ward's method [121] in which the proximity between two clusters is defined as the increase in squared error after two clusters are merged.

Agglomerative Hierarchical Clustering Algorithm

Among the agglomerative methods, perhaps the most widely used algorithm is the Agglomerative Hierarchical Clustering (AHC) algorithm [28]. In the AHC algorithm, each data point is initially considered as an individual cluster.

Then, the next step is to find the closest pair of clusters by considering the inter-cluster distance values stored in the proximity matrix, and subsequently merge the closest pair of clusters into a single cluster. At this moment, the proximity matrix is also updated to reflect the merge operation. This process is repeated until the desired number of clusters has been generated. In the AHC algorithm, once a decision has been made to merge two clusters, it is impossible to undo this operation. Also, this algorithm is very sensitive to the stopping criterion, which greatly affects the clustering results. If the stopping criterion is too strict, too many clusters will be obtained. On the other hand, if the stopping criterion is too soft, fewer number of clusters will be formed. The AHC algorithm can be found in [77].

2.3.2 Partitional Clustering Algorithms

Different from hierarchical clustering in which a set of clusters are generated in a nested structure, clusters generated by partitional clustering algorithms [9, 43] are unnested. Partitional clustering algorithms divide the data into non-overlapping subsets (so called *partitions*) such that each data point is within exactly one subset. The collection of all such clusters is called partitional clustering. One of the most widely used partitional clustering algorithms [9, 11] is the K-means algorithm.

K-means Algorithm

The K-means algorithm [33, 34] is in fact a prototype-based partitional clustering algorithm. By “prototype-based”, we mean that each data point is closer to the prototype of the cluster to which it belongs, than to the prototype of other clusters. In the K-means algorithm, the notion of prototype is represented by a centroid, which is the average data point of all data points within a cluster. The goal is to achieve a user-defined number of (K) clusters.

The K-means algorithm begins with the selection of K initial centroids, where K is the desired number of clusters. Thereafter, all the data points are assigned to their closest centroid, which forms the initial set of clusters. Next,

according to the data points assigned to each cluster, the cluster centroids are updated to reflect the average value of the data points within the corresponding clusters. This data point assignment and centroid re-calculation steps continue until centroids stabilizes (i.e., no longer changes). More details of the K-means algorithm can be found in [134].

2.3.3 Neural Network-based Clustering Techniques

Apart from hierarchical and partitional approaches, another type of clustering technique that has emerged and is of growing use is the neural network-based approaches [38, 107]. The concept of neural networks [36] (more formally, Artificial Neural Networks (ANN)) has been inspired by the human brain, which comprises closely interconnected sets of neurons. When a neural network is used for data clustering purposes, the goal is to find features in the data that characterize the desired output [27]. Kohonen's Self-Organizing Map (SOM) is a well known neural network that is commonly used for data clustering purposes [55, 72, 103].

Kohonen's Self-Organizing Map

When the SOM is used for clustering [109], similar to prototype-based clustering techniques such as K-means, data objects are processed one at a time and the closest centroid is updated. However, unlike K-means, SOM imposes a topological ordering of the centroids and neighbourhood centroids are also updated. One of the main advantages of using SOM is that it is quite robust to noisy data [130]. In addition, SOM is one of the most popular techniques for handling high dimensional data, which makes it very suitable for document clustering. For the down-side, SOM does not allow incremental learning. That is, whenever a new document is added to the document collection, the learning process needs to be performed on the entire document collection again, which makes the algorithm computationally expensive.

The goal of SOM is to represent a high dimensional input space, using a low dimensional (typically two dimensional) space called a *map*. Each node in the

map is associated with a weight vector that is of the same dimension as the input vectors (i.e., document vectors we want to cluster). Also, nodes in the map are structured such that they seek to preserve the topographic ordering of the input space. SOM basically has two modes of operation: training and mapping. In the training phase, the map is built using sample inputs. This is a competitive process, in which the output nodes compete to become the winning node (or winning neuron). When the training process is completed, the nodes in the map will be organized such that different regions of the map respond collectively to each particular input pattern. In the mapping phase, an (unknown) example is presented to the map, and SOM tries to find the best matching neuron that has a weight vector closest to the input vector. More details of the SOM neural network training algorithm can be found in [54].

2.3.4 Hybrid Clustering Techniques

One of the major issues in using a single clustering algorithm, such as the K-means algorithm, is that the quality (or “goodness”) of the clustering is subjective to the initialization parameters chosen for the clustering algorithm. Moreover, each clustering algorithm has its own advantages as well as drawbacks that affect the overall clustering result. For example, in the Agglomerative Hierarchical Clustering (AHC) algorithm, once a training sample is assigned to a certain cluster, it cannot be subsequently moved to another cluster.

As a solution to the aforesaid limitations and aiming at better clustering results, researchers have introduced approaches that combine different clustering techniques, which are usually referred to as *hybrid clustering techniques* [16, 49, 127]. One example is the two-stage clustering technique proposed by Punj and Steward [83], which combines a hierarchical clustering technique (e.g., the Ward’s minimum variance method) with a non-hierarchical clustering algorithm (e.g., K-means). The idea is that the Ward’s minimum variance method can be used to determine the candidate number of clusters and the starting points of each cluster, which can be eventually fed as input into the non-hierarchical K-means clustering technique to obtain the final clustering solution. Such approaches resolve the issues

of choosing the desired number of clusters and the cluster initialization problem of the K-means algorithm. In addition, it can lead to an improved final solution compared to using the individual clustering techniques alone.

Kohonen's Self-Organizing Map + K-means Hybrid Clustering Technique

Another variant of hybrid clustering technique has been proposed by Kuo et al. [59], by integrating the Kohonen's Self-Organizing Map (SOM) neural network with the K-means algorithm (SOM+K-means). That is, they replaced the hierarchical clustering method used for the first stage of the two-stage technique proposed by Punj and Steward [83] with the computational intelligence technique, SOM. In this approach, the SOM is trained in the first stage in order to obtain the candidate number of clusters and the starting point of each cluster. Then, in the second stage, the result of SOM is used to initialize the K-means algorithm, and the K-means algorithm is applied to obtain the final clustering. This approach has been found to perform even better than the conventional two-stage method (i.e., Ward's minimum variance method followed by K-means) [59]. Unlike the hierarchical clustering technique where cluster membership decisions are non-revocable, SOM has the capability to continually update the neighbouring observations of each cluster, which leads to a fine-tuned set of clusters. Besides, the advantages of SOM such as resilience to noisy data and the ability to order input data both locally as well as globally [40] can help in providing an optimum input for the K-means algorithm, which can aid in leading towards a better overall solution. Due to its effectiveness, this approach (or a similar approach) has also been utilized in some other projects [39].

2.4 Summary

In this chapter, we have discussed the concepts and techniques related to document retrieval. As mentioned in Chapter 1, this research focuses on applying clustering techniques for formula clustering as well as mathematical document clustering. The VSM is used for the representation of mathematical formulas and for the representation of mathematical documents in preparation for formula

clustering and mathematical document clustering, respectively. To the most of our knowledge, no existing projects have investigated clustering techniques for mathematical knowledge retrieval. Therefore, this research aims to investigate clustering techniques for both formula clustering as well as mathematical document clustering based on the four different types of clustering algorithms, AHC, K-means, SOM, and SOM+K-means.

CHAPTER 3

MATHEMATICAL DOCUMENTS

Since the primary goal of this research is about retrieving mathematical documents, a concept that is central to this research work is the notion of a mathematical document. Therefore, this chapter is dedicated to discuss about mathematical documents. We will first talk about the content format of a mathematical document as applied to this research. Afterwards, the features of mathematical documents will be discussed.

3.1 Notion of Mathematical Document

When we say a “mathematical document”, it would be possible to give different interpretations for that. For instance, in [52], a mathematical document is a web document containing any of the following types of information: formulae, mathematical statements, or large-scale theory structure. Whereas in [14], a mathematical document is a document containing one of the following information: a theorem, a definition, or a proof. Because of these differences, it is necessary to clearly define the notion of a mathematical document (i.e., the content format) as related to this research work.

As mentioned in Chapter 1, the primary goal of this research work is on retrieving mathematical documents for problem solving. In other words, mathematical documents are retrieved from a problem solving perspective. Hence, the content format of a mathematical document as related to this research is such that it can be used to achieve the goal of problem solving. Therefore, a mathematical document is defined as a composite of two parts: a question part and a solution part. Figure 3-1 shows an example of a mathematical document [100]. The question part contains a mathematical problem (i.e., question). The solution part contains the corresponding step-by-step solution to the mathematical problem given in the question part.

Question
<div> </div> <p>The figure above shows the region bounded by the curve $y = \tan x$, the x-axis and the line $x = \frac{\pi}{4}$. Use the identity $\sec^2 x = 1 + \tan^2 x$ to find, correct to 2 decimal places, the volume obtained when the region is rotated through 2π radians about the x-axis.</p>
Solution
<p>The area of each circular area generated is $\pi r^2 = \pi(\tan x)^2$.</p> <p>To find the volume obtained, we need to use definite integrals in the interval: 0 to $\frac{\pi}{4}$.</p> <p>Therefore, the volume generated is:</p> $\text{Volume} = \int_0^{\frac{\pi}{4}} \pi(\tan x)^2 dx$ <p>By using the given identity $\sec^2 x = 1 + \tan^2 x$,</p> $\begin{aligned} \text{Volume} &= \int_0^{\frac{\pi}{4}} \pi(\sec^2 x - 1) dx = \pi[\tan x - x]_0^{\frac{\pi}{4}} \\ &= \pi\left(1 - \frac{\pi}{4}\right) = \frac{3\pi^2}{4} = 7.40 \text{ units}^3 \quad (\text{Answer}) \end{aligned}$

Figure 3-1: A mathematical document.

3.2 Features of Mathematical Documents

To have a better understanding about mathematical documents, it would be necessary to be familiar with the features that constitute mathematical documents. In general, a mathematical document can consist of three basic features. They are plain text descriptions, mathematical formulas (sometimes called mathematical expressions), and diagrams (e.g., geometric figures). Each of the three features expresses a mathematical idea in a unique perspective. In the following sections, we will discuss more about these three features in mathematical documents.

3.2.1 Plain Text Descriptions

Mathematical documents can contain plain text descriptions, which are purely textual parts that are formed by sequences of ASCII characters. When applying information retrieval (IR) to mathematical documents, these textual information can be processed quite easily by using general text processing techniques that are commonly applied to text documents.

3.2.2 Diagrams

Mathematical documents may also contain diagrams (e.g., geometric figures). A diagram is an analogical representation about some data or knowledge presented by visual means [56]. They provide a convenient format in which certain information can be presented in a possibly comprehensible manner. Rather than using only plain text to describe a particular incident or concept, which can be clumsy in some cases, and also cause difficulties for users in comprehending that information, diagrams may provide a method of expressing such knowledge in an intuitive manner and help users understand the knowledge “at a glance”. Figure 3-2 shows an example of a mathematical diagram [98].

Figure 3-2: Example of a diagram.

As shown in Figure 3-2, a diagram can be made up of various kinds of elements, such as alphabetical characters, numbers, lines, and axes arrows (e.g., x-axis and y-axis) which collectively form the complete diagram. As such, the easiest and most intuitive way to represent a diagram in a document would be by constructing an image containing the diagram. From an IR standpoint, however, then it becomes a complicated task to retrieve information from the diagrams since it would involve image processing. One possible solution for this is to attach some textual tags to the diagram (or provide a description in plain text) to describe the features of the diagram, so that when processing a document, features of the diagram can be extracted from those textual tags (or plain text descriptions).

3.2.3 Mathematical Formulas

Mathematical formulas are probably the most prominent feature in mathematical documents. A mathematical formula can denote a mathematical idea in a rather concise and accurate manner. Unlike plain text descriptions, which can be represented as linear sequences of characters (i.e., in a single line) in mathematical documents, mathematical formulas typically cannot be represented as an array of characters. In other words, formulas generally consist of structured mathematical text, including special features such as mathematical symbols and operators. Figure 3-3 gives some examples to illustrate the structural nature of mathematical formulas. As shown in Figure 3-3, formulas can also be regarded as two-dimensional information objects [52].

Figure 3-3: Structural nature of mathematical formulas.

Due to the structural nature of mathematical formulas, different strategies have been introduced for the digitized representation of mathematical formulas so that formulas can be stored and manipulated in computer systems. The choice of the strategy depends on the usage context or application. Two most common approaches are the use of an image representation and using a standardized representation format.

Image Representation

A common practice that has been in use for quite some time for formula representation is through creating an image containing the formula as a static bit map (digital drawing) in Graphics Interchange Format (GIF), Portable Network Graphics (PNG), or Joint Photographic Experts Group (JPEG) format [21]. Using an image is a simple and convenient method for displaying formulas across different platforms. However, there are several disadvantages of using images.

- Image formats are haunted by many usability issues such as alignment, sizing, and colour conformity with text, and requires authors to spend more time on creation, storage, and maintenance such as editing even a simple small change.
- Extraction of semantic information from formulas (e.g., in IR applications) becomes very complex as it necessitates the use of image processing and pattern recognition techniques.
- Images cannot be easily converted into alternative output modes such as speech or Braille for use by people with disabilities.
- Images require significantly higher bandwidth to transmit, which can cause problems with accessing mathematical documents through low speed internet connections.

Standardized Representation Format

In the recent past, some standardized encoding formats (commonly known as *markup languages*) have been introduced for the representation of mathematical formulas. These encoding formats are used to describe the mathematical formulas

by textual means. That is, they can represent a formula purely by using the standard set of characters available in common computer keyboards. Some of the most popular encoding formats are Mathematical Markup Language (MathML) [7], LaTeX [79], and OpenMath [12]. This formula representation strategy tends to resolve most of the problems encountered by the other strategies. However, due to the verbosity of these representation formats and the complexity involved in creating and manipulating a formula by using the markup languages, various supporting software tools and technologies have been developed to support the creation and manipulation of formulas using these representation formats.

A brief discussion on the three popular formula encoding formats, namely, LaTeX, OpenMath, and MathML is given as follows.

- *LaTeX*. LaTeX [62] is an extensively used markup language in the scientific community for scientific document authoring. It is also perhaps one of the most popular formats that is used to encode mathematical formulas into scientific documents. LaTeX has a relatively compact representation compared to the other two formats (MathML and OpenMath). It merely addresses the formatting issues of a formula (e.g., alignment, adding space between operators, small font size for superscript) and thereby producing a visually appealing formula.
- *OpenMath*. Different from LaTeX, the OpenMath standard [12] aims at the semantic representation of a formula. A formula representation in OpenMath is based on a set of concepts that are defined in the OpenMath format's accompanying "content dictionaries".
- *MathML*. MathML [7] is the emerging standard for the representation of mathematical formulas, which has already been used by several existing projects [53, 75, 76] and is gaining an increasing attention among researchers. It was developed by the World Wide Web Consortium (W3C) [124] for the purpose of encoding mathematical notation into web documents, and is a recommendation of the W3C math working group. MathML itself comes in two flavours, namely, presentation markup and content markup. The presentation markup concerns the visual appearance of a formula (like LaTeX),

while it also represents several semantic information of a formula as well [79]. On the other hand, the content markup (like OpenMath) focuses on the functional structure and the semantic representation of a formula. MathML is increasingly being used by information retrieval (IR) applications that focus on searching mathematical formulas. Also, MathML is accompanied with several supporting software tools for the creation of formulas (e.g., WebEQ [23]), presentation of formulas (e.g., MathPlayer [22]), and processing of formulas (e.g., DOM [125, 126]), which make it more useful and applicable.

Table 3-1 presents a mathematical formula representation in the three encoding formats: MathML (Presentation Markup and Content Markup), LaTeX, and OpenMath. As shown in Table 3-1, each encoding standard has its own formal representation for the same mathematical formula. However, they all basically represent two aspects of a mathematical formula: presentation aspect (i.e., the formula's visual details) and/or content aspect (i.e., the formula's functional structure and semantic meaning).

Among the different formula representation strategies, it is apparent that the use of a standardized representation format is more advantageous from an IR point-of-view. For an IR application, the ability to extract as much semantic information as possible is of vital importance. Since MathML is a recommended standard and has also been widely adopted in many other related projects, we have chosen the MathML format (presentation markup) as the representation format of mathematical formulas in this research.

Table 3-1: Representation of a mathematical formula in three encoding formats.

		$\sqrt[3]{x+y}$
MathML	Presentation Markup	$\begin{aligned} &<math> \\ &\quad <mrow> \\ &\quad \quad <mroot> \\ &\quad \quad \quad <mrow> \\ &\quad \quad \quad \quad <mi>x</mi> \\ &\quad \quad \quad \quad <mo>+</mo> \\ &\quad \quad \quad \quad <mi>y</mi> \\ &\quad \quad \quad </mrow> \\ &\quad \quad </mrow> \\ &\quad \quad \quad <mn>3</mn> \\ &\quad \quad </mrow> \\ &\quad </mroot> \\ &\quad </mrow> \\ &</math> \end{aligned}$
	Content Markup	$\begin{aligned} &<math> \\ &\quad <apply> \\ &\quad \quad <root/> \\ &\quad \quad <degree> \\ &\quad \quad \quad <cn>3</cn> \\ &\quad \quad </degree> \\ &\quad <apply> \\ &\quad \quad <plus/> \\ &\quad \quad \quad <ci>x</ci> \\ &\quad \quad \quad <ci>y</ci> \\ &\quad \quad </apply> \\ &\quad </apply> \\ &</math> \end{aligned}$
LaTeX		$\$ \sqrt[3]{x+y} \$$
OpenMath		$\begin{aligned} &<OMOBJ> \\ &\quad <OMA> \\ &\quad \quad <OMS cd="arith1" name="root"/> \\ &\quad \quad <OMA> \\ &\quad \quad \quad <OMS cd="arith1" name="plus"/> \\ &\quad \quad \quad <OMV name="x"/> \\ &\quad \quad \quad <OMV name="y"/> \\ &\quad \quad </OMA> \\ &\quad <OMA> \\ &\quad \quad <OMI>3</OMI> \\ &\quad </OMA> \\ &</OMOBJ> \end{aligned}$

3.3 Summary

In this chapter, we have discussed the core mathematical document concept of this research. We have shown that the notion of a mathematical document may be interpreted in different ways. As applied to this research, a mathematical document is defined as an information object containing two elements: a mathematical problem and its associated solution. We have also discussed the three major features in mathematical documents, namely, plain text descriptions, mathematical formulas, and diagrams.

CHAPTER 4

FORMULA SEARCH

Mathematical formulas are probably the most significant feature in mathematical documents. Therefore, the techniques used to extract and process the semantic features in formulas have a primary importance in the overall goal of retrieving mathematical documents. This chapter discusses our proposed approach for mathematical formula retrieval. We will first discuss some of the most important considerations associated with mathematical formula retrieval. Then, the related existing systems will be discussed including their approach, strengths, and limitations. The proposed formula retrieval approach basically consists of two tasks: semantic clustering and formula matching. Next, we will present the related background information on the main concept behind our proposed formula matching approach, the *tree edit distance* technique. The overall formula retrieval approach will then be discussed in details. Finally, the performance evaluation of the proposed formula matching technique and the clustering techniques will be given.

4.1 Formula Search Considerations

An information retrieval task is typically associated with one or more important considerations. As an example, for a text document retrieval task, typical considerations include identification of keywords, synonym identification, and phrase identification. Mathematical formula retrieval is also associated with a number of important considerations. Many of them have been discussed in the literature [53, 74, 76]. To highlight, some of the most important considerations in mathematical formula retrieval are listed as follows:

- *Canonical forms.* A mathematical formula can sometimes be expressed in different equivalent ways, called *canonical forms*. For a mathematical formula search engine, identification of these canonical forms enables the retrieval of “synonymous forms” of mathematical formulas. It is analogous to the

identification of synonyms in text retrieval. This helps to improve the recall in mathematical formula retrieval.

Example formula: nC_r

Canonical forms: $nCr, \binom{n}{r}$

- *Notational Variation.* The same mathematical formula can sometimes be expressed in different notations, all of which share the same meaning. When retrieving mathematical formulas, these variants should preferably be interpreted as equivalent. That is, if a query is issued for one of the variant notations, all the different variations should be retrieved. The equivalence of variant notations is formerly referred to as *modulo α -equivalence*. Identifying notational variation basically helps to improve the recall in mathematical formula retrieval.

Example formula: $\int f(x) dx$

Equivalent variations of notations: $\int f(y) dy, \int f(u) du$

- *Sub-expression matching.* A mathematical formula can sometimes contain another formula as a sub-expression (or sub-formula). Sometimes, it would be useful to retrieve mathematical formulas by specifying only a sub-expression contained in those formulas as query. It provides flexibility to the user of a mathematical formula search engine in creating a softer query rather than a complete mathematical formula.

Example formula: $\ln[(2x+1)(x+1)]$

Sub-expressions: $(2x+1), (x+1)$

- *Element order recognition and ranking.* The order of elements in a mathematical formula attaches a certain semantic meaning to it. Any change in the element order also changes the semantic meaning of the formula. When a query formula is issued to a formula search engine, recognizing the exact order of elements in the query formula is important in order to rank the retrieval results accordingly. That means, if the result set contains a formula having the

same elements in the same order as the query formula, it must be ranked higher than any other formulas having the same elements but in different orderings. This concerns the precision in mathematical formula retrieval.

Example formula: $\sin \log \cos x$

Elements in the example formula: $\sin, \log, \cos, x$

Formulas with the same elements: (1) $\cos \log \sin x$ (2) $\log \sin \cos x$
(3) $\sin \log \cos x$

Ideal ranking for the query $\sin \log \cos x$: (3) followed by the other two formulas.

In addition to the considerations given above, there are other considerations as well, such as retrieving formulas containing certain patterns (i.e., wildcard search). A good formula retrieval technique should ideally address one or more of these considerations to ensure that it can satisfy most of the user search requirements.

4.2 Formula Search Systems

In this section, we will review some of the most popular existing mathematical formula search systems. For each system, the content format(s) used to represent/encode mathematical formulas, the techniques employed, main features, and strengths and limitations of the system are discussed.

According to our knowledge, there exists a number of mathematical information systems that allow searching for mathematical formulas. MathWebSearch [53, 70] is a web-based formula search engine that can retrieve web documents containing semantically similar mathematical formulas to a given query formula. At present, it recognizes mathematical formulas encoded in MathML [7] and OpenMath [12] formats. In the approach of MathWebSearch, it first indexes mathematical formulas extracted from online repositories (e.g., Connexions) using a technique derived from automated theorem proving called *substitution tree indexing*. Then, for querying, the system provides a web-based

query interface and uses its built-in index to search for formulas similar to a given query formula.

MathWebSearch supports modulo α -equivalence in formula retrieval. However, it cannot recognize canonical forms of mathematical formulas. The system is capable of retrieving web documents containing the query formula that appear as a sub-formula (i.e., sub-expression matching), but supporting that feature have caused an increase in the index size and an additional cost in processing. This is because substitution tree indexing does not directly support sub-term search.

The math search system developed by Youssef et al. [1, 131] as part of the larger Digital Library of Mathematical Functions (DLMF) [78] project is another mathematical formula search system. Math search augments the full-text search technology to search the DLMF's collection of mathematical content. At the time of indexing, mathematical formulas contained in DLMF, which are originally in the LaTeX [62] format, are transformed into HTML (HyperText Markup Language) and XHTML (Extensible HyperText Markup Language) and stored in the index. To issue a query, the system makes use of a query language that has a syntax similar to the LaTeX format and allows the user to perform a search in a similar manner as in text search engines.

Math search is another attempt at retrieving mathematical formulas based on semantic information. By building the search engine based on existing text search technology, it strives to gain the capabilities of text search technology with the added power of recognizing mathematical formulas. Math search supports Boolean search and field-based search (fields can be "equation", "theorem", etc.) [74]. It can also recognize the structure of a formula to a great extent [1]. According to [1, 74, 131], the system neither supports modulo α -equivalence nor identification of canonical forms of formulas. Users of the system are required to learn the system's proprietary query language syntax to properly form a query.

Mathdex [75] has adopted the same model as the math search system (i.e., generalizing the text search capabilities for searching mathematical formulas) and allows searching the larger proportion of mathematical content freely available on

the World Wide Web (WWW). Its approach is to linearize mathematical formulas as a sequence of text tokens, which are then indexed by a traditional text search engine. Mathdex indexes mathematical formulas encoded in both LaTeX and presentation MathML formats [1]. Firstly, mathematical formulas available in web documents are transformed into a common format, which is a combination of XHTML and MathML. Then, a multi-pass normalization is applied in order to overcome the encoding differences in mathematical formulas. As a result, a MathML tree structure is formed for each formula. They are eventually transformed to produce a canonical MathML representation for equivalent mathematical formulas. The query interface of Mathdex provides a graphical equation editor applet [23], which consists of templates for commonly used mathematical formulas. In addition, a text box is supplied to input additional text query terms.

Mathdex supports identification of notational variation in formula search (i.e., modulo α -equivalence). By extending the text search technology, Mathdex (like the math search project developed for DLMF) has also attempted to leverage the effectiveness and optimization of text search systems into mathematical formula searching, although the performance of the system is still subject to further evaluation [75]. The graphical query interface provides users a simple, convenient, and visually appealing querying method.

The Wolfram functions site [110] provides a formula search engine [123] that allows users to search the large online repository of mathematical formulas offered by Wolfram research. Mathematical formulas in the Wolfram's data collection are stored in different formats, such as Mathematica's [67] standard format, MathML, and American Standard Code for Information Interchange (ASCII). The search tool of this system has a menu-driven interface where the user can build a query by selecting different types of formula elements such as functions, operators, constants, and numbers using drop-down menus. In addition, the user is able to filter the query results based on different function categories such as Elementary Functions, and sort the results based on various parameters such as complexity.

The Wolfram’s formula data collection together with its querying method is able to retrieve a wide range of mathematical formulas based on semantic information. However, only basic support is provided for structure-oriented formula search and for searching formulas containing certain patterns [1]. The system can retrieve canonically equivalent mathematical formulas and also supports modulo α -equivalence.

Table 4-1: Summary of prominent mathematical formula search systems.

System	Content Format	Techniques	Query Input	Supported Features
MathWebSearch	MathML, OpenMath	Substitution tree indexing	Web interface, Common templates provided	Modulo α -equivalence, Sub-expression matching (indirectly)
Math search at DLMF	HTML+XHTML	Textualization of formulas	LaTeX like query syntax	Boolean search, Field-based search (fields are: “equation”, “theorem”, etc.), Sub-expression matching
Mathdex	LaTeX, Presentation MathML	Textualization of formulas	Graphical equation editor applet	Modulo α -equivalence, Element order recognition
Wolfram formula search	Mathematica’s standard format, MathML, ASCII	Unknown (Commercial)	Menu-driven approach	Canonical equivalence, Modulo α -equivalence

Table 4-1 gives a summary of the systems discussed above. It can be seen that many systems have used MathML as the content format for the representation of mathematical formulas. Different systems have employed different techniques for mathematical formula indexing and for query input as well. The indexing techniques used by the existing systems can be generally characterized into two types: textualization of mathematical notation, or a graph-based indexing mechanism. Each system has addressed some considerations related to formula retrieval, while leaving plenty of room for further improvements.

Different from the existing systems, this research focuses on using clustering techniques for intelligent retrieval of similar mathematical formulas. Also, unlike existing systems, a formula matching approach based on the tree matching technique called *tree edit distance* is proposed in this research work. The proposed approach is capable of recognizing the order of elements in a formula and supports modulo α -equivalence, sub-expression matching, and wildcard searching (to a certain degree) as well. In addition, this research suggests the use of a mathematical regular expressions list to detect the canonical equivalence between mathematical formulas. The graphical equation editor WebEQ [23] (similar to the one used in the Mathdex [75] system) is used to formulate a formula query, which provides a flexible, intuitive, simple, and yet expressive way to issue a user's query.

4.3 Tree Edit Distance

In computer science, a *tree* is a data structure which consists of a set of nodes and edges arranged in a hierarchical manner. The *tree edit distance* is a similarity measure that can be used to compare two trees and find how similar they are [10]. It is an extension of the *edit distance* concept used for string comparison. For two trees T_1 and T_2 that are rooted (i.e., there is exactly one node designated as the root of the tree), ordered (i.e., there is a natural ordering among the children of each node), and labelled (i.e., each node is assigned a unique symbol from a finite alphabet), the tree edit distance between the two trees is defined as the minimum cost sequence of *node edit operations* that makes the two trees isomorphic [115]. There are three fundamental node edit operations: *insert*, *delete*, and *re-label* (or *rename*). Each node edit operation can be formally represented by $u \rightarrow v$ where u and v are two non-null nodes from T_1 and T_2 respectively, or the null node (λ). The node edit operation is an insert operation if $u = \lambda$ and $v \neq \lambda$, a delete operation if $u \neq \lambda$ and $v = \lambda$, or a re-label operation if $u \neq \lambda$ and $v \neq \lambda$. Inserts and deletes are inverse edit operations. Figure 4-1 illustrates the three different node edit operations.

Figure 4-1: Node edit operations on trees. (a) A tree. (b) The tree after inserting node labelled g (By deleting node labelled g in the tree in (b), tree in (a) can be obtained). (c) The tree after renaming node labelled e to h .

More formally, tree edit distance can be further explained in terms of the cost of node edit operations as follows. Let S be a sequence $s_1, s_2, \dots, s_k$ of node edit operations that transforms T_1 into T_2 , and let the cost function $\gamma(u \rightarrow v)$ be the cost of a node edit operation $u \rightarrow v$, where $\gamma(u \rightarrow v)$ is a non-negative real number. Then, if the cost of a sequence of node edit operations S is given by $\gamma(S) = \sum_{i=1}^k \gamma(s_i)$, the tree edit distance $d(T_1, T_2)$ between T_1 and T_2 is defined as:

$$d(T_1, T_2) = \min \{ \gamma(S) \mid S \text{ is a sequence of edit operations that transforms } T_1 \text{ to } T_2 \}.$$

Trees are one of the well-studied combinatorial structures in computer science. Many application areas involving the comparison of trees using various distance measures (e.g., tree edit distance) can be encountered. For example, in molecular biology, tree edit distance can be used to compare RNA secondary structures [44, 136] to search for structures with very similar topologies. Other examples include image analysis, automatic error detection and correction in programming languages, automated theorem proving, and natural language processing [10, 37, 47, 84, 106, 136].

The tree edit distance concept can be implemented by a number of different algorithms. Tai [106] introduced the first solution to this problem. Afterwards, Zhang and Shasha [136] proposed a simpler and faster dynamic programming [19] algorithm for rooted, ordered, and labelled trees that are numbered based on the post-order numbering of trees. Klein [50] modified the algorithm by Zhang and Shasha, and produced an algorithm that has a better worst case time complexity. Since Klein's algorithm is a modification of Zhang-Shasha algorithm, both algorithms use the dynamic programming paradigm. For two trees T_1 and T_2 with m and n nodes respectively, the time complexity of Zhang-Shasha algorithm is $O(m \cdot n \cdot \min\{\text{depth}(T_1), \text{leaves}(T_1)\} \cdot \min\{\text{depth}(T_2), \text{leaves}(T_2)\})$ [10, 120]. For Klein's algorithm, the time complexity in the worst case is $O(m^2 \cdot n \cdot \log(n))$. The space complexity of both Zhang-Shasha and Klein's algorithms is $O(m \cdot n)$. Although Klein's algorithm performs better in the worst case than Zhang-Shasha algorithm, it has been suggested that the performance of each algorithm depends on the shape (which also relates with the size) of the two trees being compared [26].

4.3.1 Zhang-Shasha Algorithm

Zhang and Shasha [136] introduced an algorithm to compute the tree edit distance between rooted, labelled, and ordered trees. Recall that rooted, ordered, and labelled trees mean, there is exactly one node designated as the root of the tree, there is a natural ordering among the siblings of each node, and each node is assigned a unique symbol from a finite alphabet. They numbered the trees according to post-order traversal of trees so that subtrees are visited first before a parent node is visited. The algorithm follows a dynamic programming approach [19]. It implies that the main problem of computing the tree edit distance between two trees is regarded in terms of a series of relevant sub-problems, which are solved iteratively in order to obtain the ultimate solution to the main problem.

The input to the Zhang-Shasha algorithm is two rooted, ordered, and labelled trees T_1 and T_2 , whose nodes are ordered according to post-order numbering. Before the algorithm execution begins, two sets of values have to be

computed for each of the two trees. They are the set of key roots (KR) and the set of left-most leaf descendents (LMLD) for each tree. A KR is defined as a node in a tree that either has a left sibling or is the root of the tree. A LMLD is defined as a node in a tree that is either a leaf node or is the first child (i.e., left-most child) of a subtree of the tree. Figure 4-2 illustrates the LMLDs and KRs for a tree. As shown in Figure 4-2, the node labelled b_2 is a KR since it has a left sibling a_1 . Also, b_2 is a LMLD as well because it is a leaf node in the tree (and is the only node in the subtree rooted at b_2).

Figure 4-2: KRs and LMLDs of a tree.

A Special SubTree (SST) is defined as a subtree rooted at a KR (i.e., there is a SST for each KR). Then, for a particular SST, the LMLD of the SST denotes the first node (i.e., node with least post-order number) of it whereas the root (i.e., KR associated with the SST) of the SST denotes the last node of it. During the algorithm execution, a pair of SSTs from T_1 and T_2 (in increasing post-order numbers of KRs) is considered in turn as the relevant sub-problems, and the tree edit distance between the pair of SSTs is computed. Finally, the last SSTs would be the full trees, and at the end we have the tree edit distance between T_1 and T_2 computed. More details of Zhang-Shasha algorithm can be found in [10, 31, 136].

4.3.2 Discussion

Extensible Markup Language (XML) documents generally have a hierarchical structure [120]. Therefore, they can be typically represented as trees. As such, it is natural to make use of tree-to-tree comparison algorithms such as tree

edit distance to search for XML documents that have similar structures [133]. This idea can be further extended to applications of XML, such as MathML, so that the tree edit distance can be used to find similar mathematical formulas that are encoded in MathML. This idea leads towards our motivation to apply tree edit distance for searching mathematical formulas in this research.

In this research, we have used the Zhang-Shasha algorithm for the purpose of matching between two mathematical formulas to estimate their similarity. Our choice for using the Zhang-Shasha algorithm is mainly because, when mathematical formulas are represented in MathML form, those MathML representations tend not to be very large. As such, when the MathML representations are modelled as trees, those trees also tend to be not very large. Therefore, the Zhang-Shasha algorithm is supposed to perform efficiently when comparing mathematical formulas. In addition, the Zhang-Shasha algorithm is quite popular and has been applied to a number of application areas [25, 31, 73]. Moreover, it has not yet been explored in mathematical domain to compute the similarity between mathematical formulas.

4.4 Proposed Approach

Figure 4-3 gives our proposed approach for mathematical formula retrieval. In the proposed approach, a database of mathematical formulas is constructed first. The formula database stores a collection of mathematical formulas in the form of their Mathematical Markup Language (MathML) representation [7]. This aims for effective semantic feature extraction from the formulas. There are two markups in MathML: content markup and presentation markup. We used the presentation markup as it conveys several semantics of a mathematical formula in addition to describing its visual appearance [79].

Figure 4-3: Proposed mathematical formula retrieval approach.

The proposed approach can be explained in terms of two main processes: Training Process and Query Retrieval Process. The Training Process comprises the following two steps:

- Formula Preprocessing and Transformation; and
- Formula Clustering.

When the Training Process is completed, a knowledge model is generated for the mathematical formulas, which is then stored into the Formula Knowledge Base for subsequent use in the Query Retrieval Process.

In the Query Retrieval Process, a user constructs a mathematical formula query and issues it to the system through the retrieval client. In response, the system retrieves a set of mathematical formulas that are similar to the user's query formula. The Query Retrieval Process consists of the following four steps:

- Query Preprocessing and Transformation;
- Cluster Identification;
- Expression Tree Construction; and
- Formula Matching and Ranking.

In the following sections, we will discuss each of the abovementioned steps in the proposed approach.

4.4.1 Formula Preprocessing and Transformation

As discussed in Chapter 1 and Chapter 3, mathematical formulas contain several semantic features, that is, mathematical definitions (or mathematical meanings) which can be attached to the formulas. It is necessary to extract these semantic features and prepare the representative features in a suitable format before we can perform any clustering task on the formulas.

In this step, the stored mathematical formulas in the formula database are extracted and processed for conversion into a suitable format that can then be used in the Formula Clustering step. To do this, a Formula Preprocessing process is applied first to extract the semantic features contained in the formulas. As a result, a keyword vector is generated for each mathematical formula, which contains a set of indicative keywords representing the semantic features of the formula and a corresponding frequency of occurrence for each indicative keyword representing the occurrence of the semantic feature within the formula. Afterwards, the resultant keyword vectors are transformed into normalized vectors by applying a Transformation process, which can then be used for clustering purposes in the Formula Clustering step.

Figure 4-4 illustrates the Formula Preprocessing and Transformation step. As shown in Figure 4-4, a list of mathematical regular expressions is used to preprocess the mathematical formulas (details of which will be further explained subsequently).

Figure 4-4: Formula Preprocessing and Transformation step.

Formula Preprocessing

Mathematical formulas are represented and stored in the formula database in the form of MathML presentation markup. As such, Formula Preprocessing extracts the semantic meanings of the mathematical formulas from their MathML presentation markup representations, and generates a set of keyword vectors for the formulas. Each keyword vector will contain a set of indicative keywords that can be used to describe the semantic meanings of its representing formula with the corresponding frequency of occurrences for each keyword.

In this research, a mathematical regular expressions list is used to preprocess the formulas. As such, we construct a list of mathematical regular expressions with their corresponding semantic meanings (represented using indicative keywords). Some examples of the mathematical regular expressions are shown in Table 4-2. The entire list of mathematical regular expressions is given in Appendix A. The list has been compiled manually according to heuristic features that can be typically observed in mathematical formulas. The regular expressions

are written according to the regular expression syntax of the Java programming language.

Table 4-2: Examples of mathematical regular expressions.

Mathematical Regular Expression	Indicative Keyword	Meaning
<code><mi[^>]*sin</mi></code>	sin	The sine of an angle (used in trigonometry).
<code>(<msup>\s*<mrow>\s*<mi[^>]*sin</mi>\s*</mrow>\s*<mrow>\s*<mo>−</mo>\s*<mn[^>]*1</mn>\s*</mrow>\s*</msup>) (<msup>\s*<mi[^>]*sin</mi>\s*<mn[^>]*-1</mn>\s*</msup>) (<mi>arcsin</mi>)</code>	sininverse	The inverse trigonometric function of sine. Sometimes referred to as "arcsin".
<code><mi[^>]*>\s*sin\s*</mi>\s*<mo[^>]*>\s*⁡\s*</mo>\s*((\s*<mo>\(</mo>\s*(\s*<mo>\[</mo>\s*))\s*<mn>[\s*.]*</code>	sinnx	Any multiple angle formulae of sine. Ex:- sin2x , sin3x , sin4x , etc.

As shown in Table 4-2, the mathematical regular expressions list stores a set of regular expressions in one column and an indicative keyword corresponding to each regular expression in another column. The “Meaning” column provides a brief description about the purpose of each record and is not used in processing. During Formula Preprocessing, each mathematical formula (i.e., MathML markup representing the formula) is matched with the entire list of mathematical regular expressions, and the indicative keywords corresponding to the incident (i.e., matching) mathematical regular expressions are collected. Afterwards, the unique indicative keyword frequencies of occurrences are counted, and subsequently used to form a keyword vector for each corresponding mathematical formula.

Transformation

To improve the usefulness and representativeness of keyword vectors for the clustering process, and to enhance the training process, the resultant keyword vectors from Formula Preprocessing are eventually transformed into normalized vectors. There are two steps performed in the Transformation process, namely, TFIDF Conversion and Normalization.

- *TFIDF Conversion.* In this step, the term frequencies in keyword vectors are re-weighted using Term Frequency Inverse Document Frequency (TFIDF) scores [87], which aims to measure the importance of each keyword attached to a formula in relation to the keyword's appearance frequency in the entire collection of formulas. In the TFIDF Conversion step, the keyword vectors are transformed into TFIDF vectors, which are then passed to the Normalization step for conversion into normalized vectors.
- *Normalization.* The TFIDF vectors can have different magnitudes of length, which can cause similar formulas to not be recognized as similar. Therefore, the TFIDF vectors need to be normalized before feeding them into clustering algorithms. As such, this step normalizes the TFIDF scores in the TFIDF vectors into unit length using the Euclidean length of each vector. As a result, a set of normalized vectors are generated, which are then fed into the Formula Clustering step for the clustering of formulas, and subsequently stored into the Formula Knowledge Base for later use in the Query Retrieval Process.

Figure 4-5 shows an example of the Formula Preprocessing and Transformation step. As shown in the figure, in Formula Preprocessing, the mathematical formula (i.e., its MathML representation) is matched with the mathematical regular expressions list, and as a result, all the indicative keywords representing the semantic features of the formula are collected together with their frequencies of occurrences. Since there is a total of 107 distinct keywords extracted from the whole formula collection, each formula is represented with a vector that has 107 dimensions. The index column in the figure represents the position of each keyword in the 107-dimensional vector. The frequency count of each keyword is represented by the corresponding value of the keyword at the i^{th} index in the keyword vector. When the Transformation process is applied on the keyword vector, the keyword vector is transformed into a normalized vector.

Figure 4-5: Example of Formula Preprocessing and Transformation.

4.4.2 Formula Clustering

This research investigates on using clustering techniques for mathematical formula retrieval. The clustering algorithms explored here are Agglomerative Hierarchical Clustering (AHC), K-means, Kohonen’s Self-Organizing Map

(SOM), and a hybrid clustering technique that integrates the SOM neural network with the K-means algorithm (SOM+K-means). The algorithms need to be trained before any retrieval task can be performed. Each of the four clustering algorithms must be trained separately since each of them uses a different training procedure due to their different architectures. Next, we will discuss the training procedure for each clustering algorithm.

Agglomerative Hierarchical Clustering Algorithm

To train the Agglomerative Hierarchical Clustering (AHC) algorithm, two training parameters need to be initialized beforehand. They are the proximity calculation method and the stopping criterion (or stopping threshold). There are three basic proximity calculation methods, namely, single link, complete link, and the group average method. For the stopping criterion, either a threshold value or the desired number of clusters can be used.

In this research, we have used the group average method as the proximity calculation method. The reason is because it compromises between the advantages and disadvantages of single link and complete link methods [65], and it has been the most widely adopted method among the three proximity calculation methods [46]. The most appropriate value for the stopping threshold is a non-trivial problem [20]. If the stopping threshold is too high, only a few number of formulas will be able to form a cluster, and the number of clusters will be large. On the other hand, if the stopping threshold is too low, larger in size but fewer number of clusters will be generated. In this work, we have used the expected number of clusters as the stopping criterion, and the AHC algorithm is trained for different expected numbers of clusters with values ranging from 5 to 100 in intervals of 5 (i.e., 5, 10, 15, ..., 100).

The training algorithm for AHC can be found in [77]. At the beginning of the training, each mathematical formula is considered as a single cluster, hence the number of clusters (NOC) is equal to the number of formulas (i.e., number of normalized vectors). As such, a proximity matrix with a dimensionality of

$\text{NOC} \times \text{NOC}$ is formed and is filled with the Euclidean distances between all possible pairs of normalized vectors. During each training iteration, the proximity matrix is scanned to find the pair of clusters that has the least average pair-wise Euclidean distance between the normalized vectors in the two clusters, because we use the group average method as the proximity calculation method. Once the cluster pair is found, they are merged to form a single cluster, thereby reducing the number of dimensions in the proximity matrix by 1. The training process continues until the remaining total number of clusters is equal to the expected number of clusters specified at the beginning as a training parameter. When the training session is completed, a knowledge model is generated for the formulas, and it is saved into the Formula Knowledge Base for subsequent use in the Query Retrieval Process.

K-means Algorithm

The K-means algorithm has only a single training parameter, the initial K centroids. Various methods have been proposed in the literature for centroid initialization, and the simplest is to randomly initialize the K centroids [104]. However, the performance of the K-means algorithm is highly subjective to the selection of the initial K centroids [113]. That is, different runs of K for the same input may produce different solutions [17]. 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).

The training algorithm for K-means can be found in [134]. At the start of the training, randomly chosen K normalized vectors are selected as the K initial centroids. Using these K initial centroids as the representatives for the K initial clusters, all the remaining normalized vectors are assigned to their closest cluster based on the Euclidean distance between each normalized vector and the cluster centroids. During the training, the centroids are updated such that they represent the average characteristics of the data points (i.e., formulas) contained in their respective clusters. The algorithm progresses its execution until the centroids are no longer updated (i.e., become stable), which is the moment when the training

process for K-means completes. The resulting knowledge model is then stored in the Formula Knowledge Base.

Kohonen's Self-Organizing Map Neural Network

To train the Kohonen's Self-Organizing Map (SOM) neural network, seven training parameters have to be decided beforehand. They are the size of the SOM grid, the dimensionality (and shape) of the SOM grid, the initial learning rate, the learning rate adjusting function, the neighbourhood function, the neighbourhood updating function, and the number of training iterations. According to Kohonen [54], there are no fixed rules for setting any of these training parameters. Therefore, in this research, the SOM has been trained in the following manner.

The initial learning rate, the learning rate adjusting function, the neighbourhood function, and the neighbourhood updating function have been preset with certain functions/values. The learning rate is typically varied between 0 and 1 [108], starting from a value close to 1 and decreasing monotonically as the algorithm progresses [54]. In this work, the initial learning rate is set to 0.1, which is a typical choice that has been used in many other projects [13, 69]. An exponential decay function [35] has been used to adjust the learning rate, which is given by the following formula:

$$L(t) = L_0 \exp\left[-\frac{t}{\lambda}\right] \quad t = 1, 2, 3, \dots$$

where $L(t)$ is the learning rate at time t , L_0 is the initial learning rate, λ is a time constant, and t is the current time-step.

For the neighbourhood function, we have set it to the commonly used Gaussian function [3] which is given by the following formula:

$$\Theta(t) = \exp\left[-\frac{|r_i - r_c|^2}{2\sigma^2(t)}\right] \quad t = 1, 2, 3, \dots$$

where $|r_i - r_c|^2$ is the distance between neuron i and the Best Matching Unit (BMU) (i.e., winning neuron) c , $\sigma(t)$ is the neighbourhood radius at time t , and t is the current time-step. In addition, to determine the size of the neighbourhood (i.e., as the neighbourhood updating function) we have used the exponential decay function [85] given by the following formula:

$$\sigma(t) = \sigma_0 \exp\left[-\frac{t}{\lambda}\right] \quad t = 1, 2, 3, \dots$$

where $\sigma(t)$ denotes the neighbourhood radius at time t , σ_0 is the neighbourhood radius at time t_0 , λ is a time constant, and t is the current time-step.

After setting the aforementioned parameters, the SOM has been trained in two different ways. In the first case, the size and the dimensionality (including the shape) of the SOM grid were fixed (that is, the number of neurons is fixed), and the SOM was trained for different number of training iterations. In the second case, the number of training iterations was fixed and the SOM was trained with different number of neurons (i.e., different grid sizes and dimensionalities).

In the first way of training the SOM, the grid size, dimensionality and shape of the SOM grid were fixed, and the SOM was trained for different number of training iterations. The size of the SOM grid determines the number of neurons (i.e., number of clusters) generated, while the dimensionality and shape of the grid determines the relative locations and neighbourhood of the neurons. The choice of the best grid size and shape are non-trivial problems [54]. In this experiment, the size of the SOM grid has been decided based on the number of neurons (N) obtained according to the following heuristic formula: $N = 5 \times \sqrt{\text{number_of_samples}}$ [119]. The number of training samples is 550, therefore $N = 5 \times \sqrt{550} = 117.26 \approx 117$. Then, to obtain the dimensionality of the SOM grid, we followed the approach given in [6]. Basically, the two largest eigenvalues of the training data were calculated, and the two sides of the SOM grid were set as close as possible to the ratio between the two largest eigenvalues.

Afterwards, the lengths of the two sides were adjusted so that their product closely matches with the number of neurons (N) obtained before. This resulted a SOM grid with a dimensionality of 13×9 (which is rectangular in shape). After setting the size, dimensionality, and shape of the grid, the SOM was trained for different number of training iterations with values ranging from 100 to 1000 in intervals of 100 (i.e., 100, 200,..., 1000).

In the second way of training the SOM, the number of training iterations was preset and the SOM was trained for different numbers of neurons (N). To obtain a model with good statistical accuracy, the number of training iterations must be reasonably large. According to Kohonen [54], a “rule of thumb” for the number of iterations is at least 500 times the number of neurons. However, Kohonen also argues that there is no relationship between the number of neurons and the number of training iterations, and it should be sufficient to have a smaller number of iterations depending on the application. As such, the number of training iterations was set to 500. Afterwards, the SOM was trained for different numbers of neurons in the range from 5 to 100 in intervals of 5 (i.e., 5, 10, 15,..., 100). For each configuration, a matching dimensionality was set for the SOM grid such that the product of two sides of the grid matches the value of N (i.e., if $N = 5$, the grid dimensionality was set to 1×5 ; if $N = 10$, the grid dimensionality was set to 2×5 , etc.).

The training algorithm for SOM is given in [54]. Before the training session begins, the weights of the neurons in SOM grid are initialized with random real numbers within the interval $[0,1]$. In each training iteration, the entire set of normalized vectors is fed into the SOM network. The winning neuron for a particular input vector is determined by finding the neuron that has the smallest Euclidean distance between the input vector and the neuron’s weight vector. Once the winning neuron is found, only the weights associated with the winning neuron and its neighbourhood are adjusted to adapt to the input pattern. Whenever a new iteration begins, the neighbourhood for the current iteration is determined and the learning rate is adjusted. When the training session for SOM is completed, the

neurons in the SOM grid represent the resulting set of clusters (i.e., knowledge model), which is stored into the Formula Knowledge Base.

SOM+K-means Hybrid Clustering Algorithm

SOM+K-means is a hybrid clustering technique that integrates the SOM neural network with the K-means partitional clustering algorithm. It first executes the SOM to determine the number of clusters and the starting points of each cluster, and eventually feeds the result of SOM into the K-means algorithm to obtain the final clustering solution. This method has been originally proposed in [59, 60]. Slightly different variations of SOM+K-means technique have been proposed in previous researches [39, 58], but the overall architecture remains quite similar. Figure 4-6 shows the flow of steps in the architecture of the SOM+K-means technique developed for this research.

In the first stage, the SOM neural network is trained. It has been performed in a similar manner to that of training the SOM as a standalone clustering algorithm. As such, the SOM has been trained in two different ways: (1) with a fixed grid size (with fixed dimensionality and shape) and varying number of training iterations; and (2) with a predetermined number of training iterations and varying number of neurons. When the training session for SOM is completed, the neurons in the SOM grid represent the resulting set of clusters, and the weight vectors of the neurons represent the cluster centroids.

Next, the second stage of clustering starts by using the training result of SOM to initialize the K-means algorithm. Here, it should be noted that we have used only the neurons that represent non-empty clusters for the initialization of K centroids in the K-means algorithm. This is because an empty cluster corresponds to a neuron having a zero-valued weight vector (i.e., all values in the weight vector are zeros) and it does not carry any importance for the initialization of the K-means algorithm. Using the result of SOM for the K-means algorithm initialization not only reduces the error incurred in initial centroid selection, but also reduces the time taken for the convergence of the K-means algorithm.

Figure 4-6: The work flow of SOM+K-means hybrid clustering technique.

After initializing the K initial centroids, the K-means algorithm is trained in a similar manner to that of the standalone K-means algorithm's training process discussed before. When the K-means algorithm finishes its execution, the training session for SOM+K-means is complete and the resulting knowledge model is saved into the Formula Knowledge Base.

4.4.3 The Formula Knowledge Base

The Formula Knowledge Base stores different types of information generated in the Training Process. The stored information basically comprises the following:

- The set of normalized vectors generated at the completion of the Formula Preprocessing and Transformation step; and
- The sets of clusters (i.e., knowledge models) generated as a result of training the AHC, K-means, SOM, and SOM+K-means clustering algorithms in the Formula Clustering step.

All of the information in the Formula Knowledge Base are stored in the form of text files. Table 4-3 shows the details of the stored information including the storage format of each information.

Table 4-3: Information stored in the Formula Knowledge Base.

Information	Description	Storage Format
Normalized Vectors	The set of normalized vectors resulted from Formula Preprocessing and Transformation step. There is a single normalized vector for each formula in the training data set.	Formula_ID Dim_1 Dim_2 .. Dim_N
Cluster Centroid Vectors	The centroid vectors of the corresponding clusters generated by each clustering algorithm.	Cluster_ID Centriod_Dim_1 Centroid_Dim_2 .. Centroid_Dim_N
Formulas in Each Cluster	For the clusters generated by each clustering algorithm, the formulas contained in each cluster.	Formula_ID,Cluster_ID
Meta-Information	Meta-information: the number of formulas and the dimensionality of the normalized vectors.	Meta_info_1 Meta_info_2

4.4.4 Query Preprocessing and Transformation

Once the Training Process is completed, the stored information in the Formula Knowledge Base is then used for query processing in the Query Retrieval Process to find similar mathematical formulas to a user’s query formula. The

Query Retrieval Process begins with the Query Preprocessing and Transformation step. In this step, the user submits a query to the system using the retrieval client. The WebEQ [23] equation editor is used to formulate the formula query. The user's query formula is first preprocessed in a similar manner to the Formula Preprocessing procedure applied in the Formula Preprocessing and Transformation step in the Training Process, and it results a query vector (similar to a keyword vector). The resulting query vector is then transformed into a normalized query vector by applying a similar process to the Transformation procedure explained in the Formula Preprocessing and Transformation step in the Training Process. However, there is one difference in transforming the query vector. Instead of weighting the query terms using TFIDF, the query terms are weighted using $qf \times idf$ [117], where qf is the frequency of occurrence of an indicative keyword in the query formula and idf is the inverse document frequency of the same indicative keyword in the formula collection stored in the formula database. The reason for using $qf \times idf$ is because a keyword that occurs more frequently in the query is likely to be more important than those that occur infrequently, and those keywords that occur infrequently in the formula collection are likely to be more important than the frequent keywords.

4.4.5 Cluster Identification

After the Query Preprocessing and Transformation step, the generated normalized query vector is then used to find the clusters containing the most similar mathematical formulas. To do this, the cluster centroid vectors stored in the Formula Knowledge Base are retrieved, and the Euclidean distance $dist(\vec{q}, \vec{c})$ is calculated between the normalized query vector $\vec{q}$ and a cluster centroid vector $\vec{c}$ as follows:

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

where n is the number of dimensions in the normalized query vector (and the centroid vector) and q_i and c_i are the weights of the i^{th} element in the normalized query vector and the centroid vector respectively.

A centroid vector represents the average characteristics of the formulas contained in its represented cluster. As such, the smaller the Euclidean distance, the more likely those clusters will contain formulas with similar characteristics to the query formula. In other words, clusters that are farther would contain formulas that are not very similar to the query formula. Therefore, next, the actual mathematical formulas (i.e., MathMLs) are retrieved from the formula database starting from the nearest cluster. In this way, sets of mathematical formulas in the clusters are retrieved, which are grouped according to their respective cluster so that the proximity between each cluster (i.e., cluster centroid) and the query formula (i.e., normalized query vector) is also maintained. The Cluster Identification step is illustrated in Figure 4-7. As shown in the figure, D1, D2, and D3 represent the Euclidean distances between the normalized query vector and Cluster 1, Cluster 2, and Cluster 3 respectively.

Figure 4-7: Cluster Identification.

4.4.6 Expression Tree Construction

The result of the Cluster Identification step is a set of formulas (i.e., MathMLs) in the clusters, which will ultimately be used to produce the result set

for the user's query. When the retrieval results are presented to the user, rather than listing them in an arbitrary order, they are best presented in a ranked order according to their similarity to the user's query. For the purpose of ranking, this research makes use of the Zhang-Shasha algorithm to compute the tree edit distance between the user's query formula and formulas in the clusters. As such, the query formula and formulas in the clusters need to be transformed into tree form before they can be fed into the Zhang-Shasha algorithm for tree edit distance computation.

The purpose of this step is to transform the query formula and formulas in the clusters into tree structures known as Expression Trees. An example of an Expression Tree is given in Figure 4-8. As shown in Figure 4-8, an Expression Tree is basically a tree consisting of nodes and edges. The node labels are represented using MathML tags extracted from the MathML representation of the corresponding formula. The Expression Tree is formed using a partial set of MathML tags extracted from the MathML representation of the formula. Only a subset of tags is used because certain tags in the MathML presentation markup deal with the visual appearance of the formula rather than its semantic meaning. The Expression Tree represents the conceptual structure of the formula while also preserving its semantic meaning (defined by the MathML elements). For instance, *exponentiation* in a mathematical formula is defined by using two elements; the *base* and the *exponent* (e.g., x^2). In the example formula given in Figure 4-8, the base *sin* has an exponent of 2 and it is represented in the Expression Tree by the `<msup>` (exponentiation or superscript operation) node with two child nodes: `<mi>sin</mi>` (base) and `<mn>2</mn>` (exponent). The process of constructing the Expression Tree will be explained later. The root node is the `<math>` tag, which is the starting tag in the MathML representation for any given formula. In the Expression Tree example, there are 3 levels in the tree; Level 0, Level 1, and Level 2. The number of levels in the tree depends on the maximum nesting level determined from the MathML representation of the formula. That means, for different formulas, the number of levels can vary depending on the complexity of their corresponding MathML representation. Nodes in the tree are numbered

according to post-order traversal of trees as specified in the Zhang-Shasha algorithm.

Figure 4-8: Expression Tree representation of a mathematical formula.

The construction of an Expression Tree can be described as follows. We make use of the same example formula given in Figure 4-8 to aid in this explanation. As discussed in Section 4.1, the same mathematical formula may be expressed in different variations of notation, which ideally gives the same meaning (formally called modulo α -equivalence). For instance, the three formulas:

$\int \sin^2 u \, du$, $\int \sin^2 y \, dy$, and $\int \sin^2 x \, dx$ can be interpreted as identical, the only difference is that the variable is changed between u , y , and x . To recognize these notational variations and interpret such notationally variant formulas as equivalent, the MathML representation of the formula is examined first for any notational differences. If any notational variations are detected, the MathML representation is normalized to a common form (called the *root form*), where all variant notations are normalized into a common notation. For example, all variables with different identities such as u, y are transformed into a common identity, in terms of x . An example is illustrated in Figure 4-9.

Figure 4-9: Normalization of MathML representations into root form.

After obtaining the MathML representation in root form, the set of MathML elements in the resulting MathML representation is examined to exclude any excessive MathML elements that are not useful for the Expression Tree construction process. As such, we identify three types of MathML elements to be excluded. They are `</math>`, `<mrow>`, and `</mrow>` tags. The `</math>` tag is the closing tag associated with the `<math>` tag and is excluded because the `<math>` tag alone is sufficient to represent a node in the Expression Tree. The `<mrow>` and `</mrow>` tags are used by the MathML presentation markup to horizontally group sub-expressions in a formula and are concerned about the visual appearance of the formula. But, they are not needed in the case of the Expression Tree. The resulting set of MathML elements after applying all exclusions to the MathML elements in the root form MathML representation is shown in Figure 4-10. It should be noted that the order of the tags (i.e., MathML elements) is preserved in the resulting set of MathML elements as it was in the root form MathML representation.

MathML Elements
<code><math></code>
<code><mo>∫</mo></code>
<code><msup></code>
<code><mi>sin</mi></code>
<code><mn>2</mn></code>
<code></msup></code>
<code><mi>x</mi></code>
<code><mi>d</mi></code>
<code><mi>x</mi></code>

Figure 4-10: Set of MathML elements for Expression Tree Construction.

Next, we proceed to construct the Expression Tree using the resultant set of MathML elements. The pseudo-code for Expression Tree construction is given in Figure 4-11. In the pseudo-code, a Composite MathML Element is defined as a MathML element that can have one or more arguments (i.e., MathML elements nested under this MathML element in the original MathML representation) for it. This implies that, in the Expression Tree, the node corresponding to the Composite MathML Element can have one or more child nodes under it (which will represent the arguments of the Composite MathML Element). For example, the `<msup>` tag requires two arguments for it; the base and the exponent. The list of all Composite

MathML Elements is given in Appendix B. This list has been compiled according to the “Table of argument requirements” given in [68]. The construction of Expression Tree is illustrated in Figure 4-12.

1. Add <math> tag as the root node of the Expression Tree. Set it as current parent node.
2. Starting from the next MathML element, process the remaining MathML elements sequentially, do steps 3-6.
3. Add the current MathML element as a new node under current parent node.
4. Check if the MathML element added in step 3 is a Composite MathML Element. If it is a Composite MathML Element, consider that MathML element as the new parent node and go to step 4.1. Otherwise go to step 5.
- 4.1 Add the next MathML element as a child node under the current parent node.
- 4.2 Continue with step 4.1 until the closing tag of the MathML element corresponding to the current parent node is found.
- 4.3 Discard the closing tag MathML element.
- 4.4 Change parent to the parent node before current parent node.
5. If any MathML elements remain, go to step 3 to proceed with the next MathML element. Otherwise, go to step 6.
6. Assign post order numbers to all the nodes.

Figure 4-11: Pseudo-code for Expression Tree Construction.

The end result of the Expression Tree Construction step is an Expression Tree for the query formula (Query Expression Tree) and sets of Expression Trees for the formulas in the clusters (Formula Expression Trees). They are then passed to the Formula Matching and Ranking step in order to rank the formulas in the clusters according to their similarity to the query formula.

Figure 4-12: Expression Tree Construction.

4.4.7 Formula Matching and Ranking

When the retrieval results are presented to the user, it is best to present them in a sorted order so that the user will quickly be able to find the most relevant results to his query. Several document ranking approaches exist, the most common ones are the ranking based on TFIDF measure and ranking based on cosine similarity. However, these approaches have problems when they are used for mathematical formula ranking. For example, suppose that we use the TFIDF measure to compare a query formula $\sin \cos \log 2x$ with three formulas in the system: $\cos \log \sin 2x$, $\log \sin \cos 2x$, and $\sin \cos \log 2x$. Among the three formulas, we would be particularly interested in ranking the formula $\sin \cos \log 2x$ as the first result since it is exactly the same formula as we are querying. But, if the similarity scores are computed for the three formulas based on the TFIDF measure, all of them would yield the same similarity score since all of them have the same set of terms $\{\sin, \cos, \log, 2, x\}$ at equal frequencies. As such, it would cause the three formulas to be ranked in any arbitrary order, which is not the ideal way we would expect. Moreover, there are several other concerns related to mathematical formula retrieval as discussed in Section 4.1. For instance, a user could be particularly interested in retrieving canonically equivalent mathematical formulas within higher orders in the ranked result set. As an example, if the query formula is $\tan^{-1} x$ and it is compared with three formulas in the system: $\tan x$, $\tan 2x$, and $\arctan x$, we would prefer in retrieving $\arctan x$ ranked higher than the other two formulas. But, the existing ranking techniques such as the TFIDF measure cannot handle these considerations and is therefore not good enough to be used for ranking mathematical formulas. Because of these problems, we need a ranking technique that is well-suited for mathematical formulas.

This research proposes a new approach for mathematical formula ranking. In this step, the formulas in the clusters are matched with the query formula, and eventually ranked according to their similarity to the query formula. To enhance the accuracy of the retrieval results, there are two types of matching performed in this step as follows:

- *Tree Matching.* The Query Expression Tree is matched with each Formula Expression Tree and the tree edit distance is computed between each pair of Expression Trees. Eventually, the tree edit distance is converted into a more precise similarity measure which we call the *Normalized Tree Edit Distance*.
- *Vector Matching.* The set of normalized vectors corresponding to the formulas in the clusters are retrieved from the Formula Knowledge Base and matched with the normalized query vector resulted from Query Preprocessing and Transformation step. As a result, the cosine similarity is computed between each normalized vector and the normalized query vector.

Tree Matching and Vector Matching will yield two similarity scores for every formula in the clusters. Finally, the two similarity scores are aggregated to compute a single Combined Similarity score for each formula, which is then used in the Ranking process to rank the formulas according to their similarity to the query formula. The Formula Matching and Ranking step is shown in Figure 4-13.

Figure 4-13: Formula Matching and Ranking.

Tree Matching

During Tree Matching, the Query Expression Tree is matched with each Formula Expression Tree in the clusters and the tree edit distance is computed between pairs of Expression Trees. The Zhang-Shasha algorithm is used to compute the tree edit distance. To apply the Zhang-Shasha algorithm between two Expression Trees, two sets of values need to be computed for each Expression Tree beforehand. They are: (1) the set of left-most leaf descendents (LMLDs); and (2) the set of key roots (KRs). Figure 4-14 shows two Expression Trees (which we will use for the rest of the discussion) with their LMLDs and KRs highlighted.

Figure 4-14: Two Expression Trees and their LMLDs and KRs.

After calculating the LMLDs and KRs, the two Expression Trees are fed into the Zhang-Shasha algorithm together with the calculated LMLDs and KRs in order to compute the tree edit distance between them. The general idea of tree edit distance computation according to the Zhang-Shasha algorithm is illustrated in a step-by-step manner in Figure 4-15, which shows the steps involved in transforming Expression Tree 1 into Expression Tree 2.

Figure 4-15: Tree edit distance computation between two Expression Trees.

Recall from Section 4.3 that the tree edit distance between two trees is the minimum cost sequence of node edit operations that transforms one tree into the other. As such, Figure 4-15 shows the minimum cost sequence of node edit operations that transforms Expression Tree 1 into Expression Tree 2. In the example shown, two node edit operations have been applied: *insert* and *rename*, which are shown with the following functions respectively:

- Insert (*new_node_label*, *KR_post_order#*, *left_sibling_post_order#*) – Insert a new node labelled *new_node_label* as a child under the KR with post-order number *KR_post_order#* and at the position next to the sibling with post-order number *left_sibling_post_order#* into the Expression Tree.
- Rename (*existing_node_label*, *existing_node_post_order#*, *new_node_label*) – Change the label *existing_node_label* of the node with post-order number *existing_node_post_order#* into the node label *new_node_label* in the Expression Tree.

In addition, if a *delete* node edit operation is applied, it can be represented with the following function:

- Delete (*existing_node_label*, *existing_node_post_order#*) – Delete the node labelled *existing_node_label* with post-order number *existing_node_post_order#* from the Expression Tree.

Furthermore, recall from Section 4.3.1 that a Special SubTree (SST) is a subtree rooted under a KR. When the Zhang-Shasha algorithm is applied between Expression Tree 1 and Expression Tree 2, a pair of SSTs (in the increasing post-order numbers of the corresponding KRs) from the two Expression Trees is considered in turn and the tree edit distance between the two SSTs is computed. When considering the SSTs in the two Expression Trees in post-order, the first difference between two SSTs occurs between the SST rooted at KR with post-order number 5 in Expression Tree 1 (consists of only one node: $\langle 2 \rangle$) and SST rooted at KR with post-order number 5 in Expression Tree 2 (consists of only one node: $\langle 5 \rangle$). To make the two SSTs identical, the node labelled

$\langle mn \rangle 2 \langle /mn \rangle$ in Expression Tree 1 should be changed (i.e., renamed) into $\langle mn \rangle 5 \langle /mn \rangle$. Afterwards, the next difference between two SSTs occurs between the SST rooted at the root of Expression Tree 1 and the SST rooted at the root of Expression Tree 2. To make the two SSTs identical, two nodes with node labels $\langle mo \rangle \− \langle /mo \rangle$ and $\langle mn \rangle 2 \langle /mn \rangle$ must be inserted in the same order into Expression Tree 1 under its root. It should be noted that whenever a new node is inserted into an Expression Tree (i.e., *insert* operation) or an existing node is deleted from the Expression Tree (i.e., *delete* operation), nodes in the Expression Tree are verified to check whether proper post-order numbering has been maintained, and if necessary, the nodes are re-assigned with correct post-order numbers. For a *rename* operation, the post-order number does not change.

As a result of applying the Zhang-Shasha algorithm, three node edit operations (i.e., two *insert* operations and one *rename* operation) are used to transform Expression Tree 1 into Expression tree 2. In this research, we assume that each node edit operation has a unit cost. Therefore, the tree edit distance between the two Expression Trees is 3 (i.e., $2+1$), which is the minimum cost sequence of node edit operations that transforms Expression Tree 1 into Expression Tree 2.

The outcome of matching between the Query Expression Tree and each Formula Expression Tree is a set of tree edit distance values obtained for each formula in the clusters. Here it is worthwhile to note that two Expression Trees being compared can be of different complexities, meaning that they can be of different orders of magnitude in terms of their breadth and depth. This may result in the tree edit distance value not correctly reflecting two similar formulas that are similar. The differences in complexity of Expression Trees can be caused by different factors such as the number of and type of symbols used in a formula and the complexity of the formula, which causes the resulting MathML representation for the formula to grow in length and depth (i.e., nesting levels), thus affecting its corresponding Expression Tree. An example of this kind of situation is illustrated in Figure 4-16.

Figure 4-16: Two similar formulas having distinctive Expression Trees.

As shown in Figure 4-16, although Formula 1 and Formula 2 are semantically quite similar to each other, lengths of their MathML representations are quite different. Since the Expression Trees for the formulas are constructed based on their MathML representations, the two Expression Trees corresponding to the two formulas have also become quite distinctive. As a result, the tree edit distance between the two formulas is 4, which does not reflect the similarity between the two formulas.

Hence, to address this limitation, each tree edit distance value is further normalized to a more precise similarity score called *Normalized Tree Edit Distance*, by applying a similar idea to that given in [2, 86, 116] as follows:

$$NTED(tree1, tree2) = 1.0 - \frac{TD(tree1, tree2)}{\max(|mml_form1|, |mml_form2|)}$$

where $tree1$ and $tree2$ denote the two Expression Trees being matched, $NTED(tree1, tree2)$ is the Normalized Tree Edit Distance between $tree1$ and $tree2$, $TD(tree1, tree2)$ is the tree edit distance between $tree1$ and $tree2$, and $|mml_form1|, |mml_form2|$ are the string lengths of the MathML representations corresponding to Formula 1 and Formula 2 respectively. The Normalized Tree Edit Distance ranges between 0 and 1. The higher the Normalized Tree Edit Distance, the more similar the two formulas would be. For the two formulas given in Figure 4-16, their Normalized Tree Edit Distance can be calculated as follows:

$$TD(tree1, tree2) = 4; |mml_form1| = 128; |mml_form2| = 200;$$

$$\therefore NTED(tree1, tree2) = 1.0 - \frac{4}{200} = 0.98$$

The similarity score of 0.98 provides a clearer indication of the similarity between the two formulas compared to their tree edit distance. This is also because tree edit distance does not have a clear scale from which a decision can be derived. Therefore, the tree edit distance values obtained for each formula in the clusters are replaced with their respective Normalized Tree Edit Distance values.

The outcome of Tree Matching is a set of Normalized Tree Edit Distance values obtained for each formula in the clusters. These values are retained as a partial similarity score for each formula that will later be used in calculating the ultimate similarity score for each formula.

Vector Matching

The tree edit distance technique used in Tree Matching is able to uncover most of the similarities between features of two formulas. However, there is one problem. It cannot recognize the canonical equivalence between two mathematical formulas. For instance, consider the same example explained before, a query formula $\tan^{-1} x$ is compared with three formulas in the system: $\tan x$, $\tan 2x$ and $\arctan x$. We would ideally prefer to have $\arctan x$ retrieved and ranked as the first result (or ranked higher than the other two formulas) in the result set since it gives

exactly the same semantic meaning as $\tan^{-1} x$. To achieve this, the MathML representations corresponding to the two formulas should be identical so that the two Expression Trees corresponding to the two formulas would also be the same, causing the tree edit distance between the two formulas to be zero. But, as shown in Figure 4-17, the MathML representation of $\arctan x$ and $\tan^{-1} x$ are different, which means the two formulas would be recognized as dissimilar and hence can cause $\arctan x$ to be ranked undesirably low.

Figure 4-17: Two canonically equivalent mathematical formulas having different MathML representations.

Vector Matching is used to solve this problem. It calculates the cosine similarity between normalized vectors corresponding to each formula in the clusters and the normalized query vector. The whole idea is as follows. The normalized vectors and the normalized query vector are generated based on the features extracted through regular expression matching (i.e., by using the list of mathematical regular expressions). When the mathematical regular expressions list is constructed, it is done in such a manner that semantically similar formulas (i.e., their MathML representations) are recognized by mathematical regular expressions that are associated with the same indicative keyword. That means, for instance, the mathematical regular expressions that match with $\arctan x$ and $\tan^{-1} x$ are described using the same indicative keyword “taninverse”. As such, the feature vectors generated for the two formulas will contain identical values for the

corresponding dimensions, which also cause the consequent normalized vectors to contain the same values in their corresponding dimensions. Therefore, when the cosine similarity is calculated between the two normalized vectors, it will yield a similarity score of 1.0, which means that the two formulas are recognized as identical. This idea also holds for partial canonical equivalence between formulas. For instance, the two formulas $\log_e(2x+1)$ and $\ln(5x+1)$ are not fully semantically equivalent although they have some partial canonical equivalence (e.g., between $\log_e$ and $\ln$). For such cases, the cosine similarity would still yield a higher value to demonstrate their canonical equivalence. Some additional examples of canonical equivalences in formulas are listed in Table 4-4.

Table 4-4: Canonical equivalences in formulas.

Example	Canonical Forms	Description
nC_r	$nCr, \binom{n}{r}$	Combination.
$\sqrt{x+1}$	$(x+1)^{\frac{1}{2}}$	Square root.
$f'(x)$	$\frac{dy}{dx}$	First derivative.
$\times$	$\cdot$	Multiplication.

In Vector Matching, the normalized vectors stored in the Formula Knowledge Base are retrieved and matched with the normalized query vector. As a result, the cosine similarity is calculated for each matching. The result is a set of cosine similarity scores obtained for each formula in the clusters. They are retained as a partial similarity score for each formula that will subsequently be used to calculate an overall similarity score for each formula.

Calculate Combined Similarity

In this process, the partial similarity scores that have been calculated for each formula in the clusters during Tree Matching and Vector Matching are merged into a single Combined Similarity score, which will eventually be used in the ranking process. The formula used to calculate the Combined Similarity is given as follows:

$$\text{Combined_Similarity} = (\text{cosine_similarity} \times wf) + \text{Normalized_Tree_Edit_Distance}$$

As shown in the abovementioned formula, a weight factor wf is used to adjust the weightage given to the cosine similarity. Here, we assume that the Normalized Tree Edit Distance has a higher significance than the cosine similarity. The exact value for wf has been derived through a series of trial experiments which gives the best overall accuracy for formula matching. The experimental results are discussed later in the performance evaluation section under the discussion of formula matching accuracy evaluation.

The result of the Calculate Combined Similarity process is a Combined Similarity score calculated for each formula in the clusters, which are then passed into the Ranking process to rank the formulas accordingly.

Ranking

During Ranking, mathematical formulas in the clusters are ranked according to their Combined Similarity scores. Each cluster is considered in turn according to its closeness to the query formula (i.e., normalized query vector) identified during the Cluster Identification step, and formulas in the cluster are ranked according to the Combined Similarity score of each of them. As a result, a ranked list of formulas is returned to the retrieval client and displayed to the user as the result set.

4.5 Performance Evaluation

The performance evaluation has been conducted using a prototype system (a screenshot is shown in Figure 4-18) which we have developed according to our proposed approach. In all, the training efficiency of the clustering algorithms, the matching accuracy of the formula matching technique, and the retrieval performance for the different clustering algorithms have been evaluated. The experiments were conducted on an Intel Core 2 Duo IV 1.86 GHz workstation with 3.25 GB of RAM running Windows XP operating system.

Figure 4-18: Screenshot of the formula search engine prototype.

4.5.1 Experimental Setup

Before discussing the experimental results, it would be necessary to have a clear idea about the data set used for the evaluations and the different types of performance measures taken during the evaluations.

Data Set

In order to conduct the evaluation of a mathematical formula search engine in an effective manner, it is important to select a data set that is representative of different types of semantic features that are typically encountered in mathematical formulas. This is because we need to verify the effectiveness of the retrieval engine in identifying the different types of semantic features contained in the formulas.

To conduct this evaluation, a data set consisting of 550 mathematical formulas has been used. The formulas in the data set are selected from the Integration (or Integrals) topic in mathematics. The main reasons behind using Integration formulas are, they contain: (1) a great amount of semantic features; and (2) a blend of features from other topical areas in mathematics as well. For instance, consider the Integration formula: $\int_0^{\pi} (x^2 + 1) \cos x \, dx$. It contains semantic features such as the Integral sign ($\int$), a numerical lower limit value ($\int_0$), a non-numeric upper limit value ($\int^{\pi}$), the special mathematical constant π (π), a polynomial expression $(x^2 + 1)$ (algebraic feature), power (in x^2), and the trigonometric function "cos" (trigonometric feature). As such, the formula contains many different semantic features including features from the formulas in topics other than Integration as well. Hence, it signifies that Integration formulas are generic enough to test our formula search engine.

Table 4-5 shows the composition of the formula data set in relation to the key semantic features contained in the formulas [111]. As shown in the table, the entire formula data set can be realized in terms of 6 different categories based on the semantic features of the formulas. It should be noted that formulas that are Definite Integrals have also been classified based on the features of the main function to be solved. For example, the formula $\int_1^3 \frac{6}{x^2} \, dx$ shown in Table 4-5 has been categorized under Rational Functions, since the main function to be solved is $\frac{6}{x^2}$, which is a rational function.

Performance Measures

In this evaluation, basically three different dimensions of performance have been taken into account. They are the training efficiency, the training retrieval performance, and the testing retrieval performance.

Table 4-5: Composition of the Integration formula data set.

Formula Category	Example Formulas	Feature Description
Polynomial Functions	$\int (3x^2 + 2x + 1) \, dx$, $\int_0^2 x(x^2 - 2) \, dx$	Polynomial expressions that are non-rational and non-irrational functions.
Rational Functions	$\int \frac{1}{(x-10)^7} \, dx$, $\int_1^3 \frac{6}{x^2} \, dx$	Rational functions (i.e., Polynomial functions that can be written as a ratio of two polynomial factors).
Irrational Functions	$\int \sqrt{x+1} \, dx$, $\int_0^4 \sqrt{3x} \, dx$	Irrational functions (i.e., Polynomial functions that cannot be expressed as a ratio of two polynomials).
Trigonometric Functions	$\int \sin^2 x \, dx$, $\int x \cos x \, dx$	Trigonometric functions.
Logarithmic and Exponential Functions	$\int x \ln x \, dx$, $\int e^{2x+1} \, dx$	Logarithmic and exponential functions.
Mixed Features	$\int \ln x \tan x \, dx$, $\int e^x \sin x \, dx$	Any combination of trigonometric, logarithmic, and exponential functions.

The training efficiency has been measured for each of the four clustering algorithms, AHC, K-means, SOM, and SOM+K-means. It was measured based on the time taken to complete one training session for each clustering algorithm. Each of the four clustering algorithms has been trained using a different set of training parameters, and therefore the total training time of each algorithm basically depends on its training parameter configuration. Therefore, the training efficiency has been measured for different training parameter configurations for each clustering algorithm.

As part of the retrieval performance evaluation, we have basically measured two different aspects of retrieval performance, namely, training retrieval performance and testing retrieval performance. The training retrieval performance measures the accuracy of retrieval when the query formula already exists in the training data set. Since an existing formula is being searched in this case, the training retrieval performance focuses more on finding an exact match. On the other hand, testing retrieval performance measures the accuracy of retrieval when the query formula does not exist in the training data set. But at least one or more

similar formulas to the query formula can be found from the training data set. In other words, the testing retrieval performance measures the generalization capability of the formula search engine beyond the training data.

4.5.2 Training Efficiency

As mentioned before, the training efficiency has been measured based on the time taken to complete one training session for AHC, K-means, SOM, and SOM+K-means clustering algorithms. Each of the four clustering algorithms has a different set of training parameters, and the total training time of each particular algorithm basically depends on its training parameter configuration. Therefore, the training efficiency has been measured for different training parameter configurations for each clustering algorithm. The selection of the training parameters for each clustering algorithm has been discussed in Section 4.4.2. Next, we will discuss the training efficiency of each clustering algorithm.

Agglomerative Hierarchical Clustering Algorithm

Figure 4-19: Training efficiency of AHC.

The training efficiency of the Agglomerative Hierarchical Clustering (AHC) algorithm for different expected numbers of clusters (i.e., different stopping criterions) is shown in Figure 4-19. The horizontal axis is divided into equal-sized intervals of 5 for the number of clusters ranging from 5 to 100. The vertical axis represents the total training time. The curve has been obtained by interpolating the total training time for different number of clusters. It can be observed that the total training time increases sub-linearly with the desired number of clusters.

K-means Algorithm

Figure 4-20: Training efficiency of K-means.

Figure 4-20 shows the total training time taken for different values of K for the K-means algorithm. The curve has been obtained by interpolating the total training time for different values of K . As shown in the figure, when the number of clusters increases, the total training time tends to increase exponentially.

Kohonen's Self-Organizing Map Neural Network

As discussed in Section 4.4.2, the Kohonen's Self-Organizing Map (SOM) has been trained in two different ways. Figure 4-21 shows the total training time

spent on different configurations of training iterations according to the first method of training the SOM. The curve has been obtained by interpolating the total training time for different numbers of training iterations. It can be observed that the total training time increases linearly with the number of training iterations.

Figure 4-21: Training efficiency for different numbers of training iterations configured on SOM.

The training efficiency of SOM based on the different numbers of neurons configured on the grid according to the second method of training the SOM is shown in Figure 4-22. The curve has been obtained by interpolating the total training time for different numbers of neurons (i.e., different numbers of clusters) configured on the SOM grid. It can be seen that the total training time increases as the number of neurons is increased.

Figure 4-22: Training efficiency for different number of neurons configured on SOM.

SOM+K-means Hybrid Clustering Technique

As discussed in Section 4.4.2, the SOM+K-means clustering technique has been trained in two different methods decided by the SOM: (1) with a fixed grid size (and shape) and varying numbers of training iterations; and (2) with a predetermined number of training iterations and varying numbers of neurons. In the SOM+K-means technique, after training the SOM, the K-means algorithm is trained using the output produced by SOM.

Figure 4-23 shows the training efficiency of SOM+K-means for different numbers of training iterations configured on SOM (according to the first method of training the SOM). The curve has been obtained by interpolating the total training time for different numbers of training iterations configured on SOM. It can be observed that the total training time increases linearly with the number of training iterations configured on SOM.

Figure 4-23: Training efficiency of SOM+K-means for different numbers of training iterations configured on SOM.

Figure 4-24: Training efficiency of SOM+K-means for different numbers of neurons configured on SOM.

The training efficiency of SOM+K-means technique based on the different numbers of neurons configured on SOM (according to the second way of SOM configuration) is shown in Figure 4-24. The curve has been obtained by

interpolating the total training time for different numbers of neurons configured on SOM. It can be seen that the total training time increases linearly with the number of neurons configured on SOM.

In summary, the total training time of AHC and K-means clustering algorithms are subjective to the number of clusters generated. For SOM and SOM+K-means clustering techniques, the training time increases linearly with the number of neurons and the number of training iterations configured on SOM. AHC has shown better training efficiency compared to the other clustering techniques when the number of clusters increases. For the K-means algorithm, an exponential growth in the training time can be observed as the number of clusters is increased. According to the results obtained, although the training time for higher degrees of configuration (e.g., larger number of clusters, higher number of training iterations) may be quite high, the necessity of using such a configuration is subjective to the impact of the resulting model on the retrieval performance. Furthermore, in the case when the training process is conducted offline, the training time will not affect the online retrieval performance.

4.5.3 Retrieval Performance

The retrieval performance can be evaluated based on the retrieval efficiency and the retrieval accuracy (or precision). Retrieval efficiency can be measured by the online retrieval speed, which is the latency in obtaining results once a user has submitted a query. There are different metrics that can be used to measure the retrieval accuracy. In this evaluation, we have used the metric “Average Precision at Seen Relevant Documents” [129] to evaluate the retrieval accuracy. This metric takes into account the precision of retrieved results as well as their rank order in the result set. It favours the retrieval of relevant documents earlier, i.e., higher in ranking. Since users typically spend more time on examining the top results than lower ranked ones [93] when using a search engine, this measure happens to be a good usability indicator of the retrieval engine.

When using the Average Precision at Seen Relevant Documents metric to evaluate the retrieval accuracy, precision values are calculated whenever a new relevant result is seen in the result set. For example, suppose for a given query, formulas ranked at 1, 3, and 6 are the only ones that are found to be relevant (i.e., other formulas are irrelevant). Then, the retrieval precision for that query according to the Average Precision at Seen Relevant Documents metric is calculated as follows:

$$\text{Retrieval Precision} = \frac{\frac{1}{1} + \frac{2}{3} + \frac{3}{6}}{3} = 0.72$$

In addition, to evaluate the retrieval precision, it is required to clearly define the notion of relevance. According to Harter [32] and Saracevic [90], there are two main classes of relevance: objective relevance and subjective relevance. Objective relevance, also called topical relevance, is defined as a relevance measure such that a document is relevant if it deals with the same topic as the request [32]. On the other hand, subjective relevance is concerned with the appropriateness of a retrieved information object relative to the information request, and is based on the intellectual interpretations carried out by human observers [89]. That is, subjective relevance is solely dependent on the human judgement of whether a retrieved document can satisfy his/her information need.

In our experiments, we have used subjective relevance in evaluating the retrieval precision. This is mainly because, subjective relevance takes into account the user's knowledge and actual needs, which considers the usefulness of a retrieved information object in meeting the user's information need [41], whereas objective relevance does not consider the appropriateness of a retrieved information object in meeting the user's needs and contexts in which queries are submitted [15]. For each query, we have manually analyzed the retrieved result set for relevant mathematical formulas.

According to our knowledge, currently there does not exist any standard means to evaluate the relevance of a given mathematical formula in relation to

another formula. Therefore, in this evaluation, the relevance of a retrieved result (i.e., formula) has been determined based on the similarity between features of the query formula and the retrieval result. For example, if the query formula is $\int x^2 \sin x \, dx$, formulas such as $\int x \sin x \, dx$ and $\int x^3 \sin x \, dx$ are considered as relevant.

During the retrieval performance evaluation, a set of search queries are issued to the system. For each search query, the retrieved formulas in the result set are analyzed to assess whether they are relevant to the query formula. As the returned list of formulas can be huge, it is difficult to examine each and every formula. Therefore, only the top 20 ranked formulas in the result set were assessed during each search session. The remaining results were considered irrelevant. If the returned list contains less than 20 formulas, all formulas were examined. The relevance was assessed using two categories: relevant or not relevant. The values of 1 and 0 were assigned respectively to the abovementioned categories.

Also, in this evaluation, the following methods have been used to calculate the average online retrieval speed and average retrieval precision. The average online retrieval speed is obtained by averaging the retrieval speed of individual search queries tested on the particular search session. The average retrieval precision for a search session is obtained by averaging the retrieval precision scores obtained for individual search queries tested in that search session.

As part of the retrieval performance, basically three different evaluations have been carried out. They are:

- (1) Matching accuracy (of the formula matching technique);
- (2) Training retrieval performance (for the clustering techniques); and
- (3) Testing retrieval performance (for the clustering techniques).

Matching Accuracy

In this evaluation, the matching accuracy of the formula matching technique has been evaluated. The idea here is to measure the effectiveness of the matching technique to observe how accurately it can identify different features in mathematical formulas and also recognize similar formulas. The accuracy has been measured using the Average Precision at Seen Relevant Documents metric.

To conduct this evaluation, a subset of formulas consisting of 130 formulas (listed in Appendix C) was extracted from the entire formula data set. The selected formulas were clustered manually into 6 clusters based on their features. Two sets of test queries were designed to evaluate the training retrieval performance (30 queries) and testing retrieval performance (30 queries) (listed in Appendix D). For this evaluation purpose, the training and test queries were categorized based on the features against which they are tested upon.

Recall from Section 4.4.7 (Formula Matching and Ranking step), a weight factor wf has been declared in the calculation of Combined Similarity score for a formula, that is used to adjust the weightage given to the cosine similarity. This weight factor needs to be evaluated first prior to the evaluation of matching accuracy and also prior to the retrieval performance evaluation of any clustering algorithms. To do this, an experiment was conducted using the same subset of formulas and the same set of test queries that was designed for the evaluation of matching accuracy, and the training retrieval performance and testing retrieval performance were evaluated for different values of wf . Table 4-6 shows the results of this experiment.

As shown in Table 4-6, the best combination of training retrieval performance and testing retrieval performance is obtained when the value of wf is 0.008, 0.009, and 0.01. Thus, we chose the value of wf to be equal to 0.009.

With the chosen value for wf , the matching accuracy was evaluated. During the evaluation, each cluster was considered in isolation and the training

retrieval performance and testing retrieval performance were evaluated using the set of designated test queries for the respective clusters. The experimental results are shown in Table 4-7. As shown in the table, our formula matching technique has achieved an overall 100% training retrieval performance and around 96% testing retrieval performance in terms of matching accuracy.

Table 4-6: Experimental results for the evaluation of wf .

wf	Training Retrieval Performance (Accuracy)	Testing Retrieval Performance (Accuracy)
0.001	100%	83.28%
0.002	100%	85.83%
0.003	100%	88.7%
0.004	100%	91.03%
0.005	100%	92.5%
0.006	100%	92.5%
0.007	100%	94.16%
0.008	100%	96.67%
0.009	100%	96.67%
0.01	100%	96.67%
0.02	100%	95.47%
0.03	100%	95.41%
0.04	100%	93.66%
0.05	100%	91.13%
0.06	100%	87.8%
0.07	100%	87.63%
0.08	100%	87.63%
0.09	100%	87.44%
0.1	100%	87.44%

Table 4-7: Matching accuracy results.

Cluster	Matching Accuracy (Training Retrieval Performance)	Matching Accuracy (Testing Retrieval Performance)
Polynomial and Power Functions	100%	100%
Rational Functions	100%	90%
Irrational Functions	100%	100%
Trigonometric Functions	100%	100%
Logarithmic and Exponential Functions	100%	100%
Mixed Features	100%	86.67%
<i>Average</i>	100%	96.11%

Training Retrieval Performance for Clustering Techniques

The training retrieval performance has been evaluated for AHC, K-means, SOM, and SOM+K-means clustering algorithms. Here we have used the same set of test queries that was used in evaluating the training retrieval performance during the evaluation of matching accuracy (listed in Appendix D). The training procedure for each algorithm was discussed in Section 4.4.2. According to the set of training parameters configured for the particular training session, the training retrieval performance was evaluated.

In summary, all of the four clustering techniques (AHC, K-means, SOM, and SOM+K-means) were able to reach a maximum of 100% training retrieval performance. In fact, the system was generally able to retrieve the formula being queried as the top result. The conclusion is that all of the four clustering algorithms are equally capable of retrieving an existing formula in the data set if the same formula is queried. A summary of the best training retrieval performance scores for the four clustering algorithms are shown in Table 4-8.

Table 4-8: Summary of best training retrieval performance for AHC, K-means, SOM, and SOM+K-means clustering techniques.

<i>Clustering Technique</i>	Best Training Retrieval Performance
AHC	100%
K-means	100%
SOM	100%
SOM+K-means	100%

Testing Retrieval Performance for Clustering Techniques

It is typical that the search queries may consist of formulas that cannot be exactly matched in the training data set. Therefore, it is important to evaluate the testing retrieval performance to verify the generalization ability of the retrieval engine beyond training data. To do this evaluation, the same set of test queries that was used in evaluating the testing retrieval performance during the evaluation of matching accuracy has been used (listed in Appendix D).

The average retrieval precision scores obtained for AHC and K-means algorithms are compared in Figure 4-25. The horizontal axis denotes the number of clusters tested on a particular search session. K-means achieved the best testing retrieval performance among the two algorithms. Also, it can be observed that the overall performance of the K-means algorithm is better than AHC. In addition, both AHC and K-means achieved their best testing retrieval performance scores when the number of clusters is not very significant (e.g., less than 50).

Figure 4-25: Testing retrieval performance for AHC and K-means.

Figure 4-26 shows the average retrieval precision values obtained for SOM and SOM+K-means clustering techniques in relation to the number of training iterations configured on SOM. It can be noted that, in comparison to the results of AHC and K-means, SOM and SOM+K-means have demonstrated only an average testing retrieval performance of around 80%-88% as the number of training iterations was varied. In other words, no significant improvement could be noted on the testing retrieval performance with different number of training iterations.

Figure 4-26: Testing retrieval performance for SOM and SOM+K-means according to the number of training iterations configured on SOM.

Figure 4-27: Testing retrieval performance for SOM and SOM+K-means according to the number of neurons configured on SOM.

The testing retrieval performance for SOM and SOM+K-means according to the number of neurons configured on SOM are compared in Figure 4-27. The best performance was achieved by SOM+K-means when the number of neurons in

the SOM grid is 40. However, both techniques have been able to achieve over 93% retrieval performance. It can be observed that the best results for both clustering techniques have been achieved when the number of neurons configured on SOM grid is not very high (e.g., less than 50). Moreover, when comparing these results to the results obtained by varying the number of training iterations (shown in Figure 4-26), it can be noted that the number of neurons configured on the SOM grid has shown more impact on the testing retrieval performance than the number of training iterations configured on SOM.

Table 4-9 gives a summary of the best testing retrieval performance scores obtained for the AHC, K-means, SOM, and SOM+K-means clustering techniques. The average online retrieval speed corresponds to the respective training parameter configuration shown. K-means and SOM+K-means have achieved the highest testing retrieval performance scores.

Table 4-9: Summary of best testing retrieval performance for AHC, K-means, SOM, and SOM+K-means clustering techniques.

<i>Clustering Technique</i>	Training Parameter Configuration	Average Online Retrieval Speed	Best Testing Retrieval Performance
AHC	Number of Clusters = 5	1.7 sec	95.83%
K-means	Number of Clusters = 30	1.56 sec	96.67%
SOM	Number of Neurons = 20, Number of Iterations = 500	1.6 sec	93.33%
SOM+K-means	Number of Neurons (SOM) = 40, Number of Iterations (SOM) = 500	1.5 sec	96.67%

In conclusion, all of the four clustering techniques have been able to uncover most of the semantic similarities between mathematical formulas as revealed by their best results of training retrieval performance and testing retrieval performance scores. Even when the number of clusters (i.e., number of neurons configured on SOM for SOM and SOM+K-means clustering techniques) is not very large (e.g., less than 50), very good testing retrieval performance have been achieved.

4.6 Summary

In this chapter, our proposed approach for mathematical formula retrieval has been presented. Basically, the use of clustering techniques for semantic clustering of similar mathematical formulas and a new formula matching approach, which is mainly based on the tree matching technique called *tree edit distance*, has been explored and discussed. For formula clustering, four different types of clustering techniques have been investigated, namely, AHC (hierarchical clustering), K-means (partitional clustering), SOM (neural network-based clustering), and SOM+K-means (hybrid clustering).

The proposed approach has been explained in terms of two processes: Training Process and Query Retrieval Process. In the Training Process, a Formula Knowledge Base is generated for the mathematical formulas. In the Query Retrieval Process, the stored information in the Formula Knowledge Base is used to find similar mathematical formulas to a user's query formula.

The performance evaluation has been conducted to evaluate the matching accuracy of the formula matching mechanism, and the training efficiency and retrieval performance of the clustering algorithms. The experimental results have shown that the proposed formula matching technique is able to achieve about 96% accuracy in terms of testing retrieval performance. The proposed formula matching technique is a combination of two types of matching: Tree Matching (which yields a Normalized Tree Edit Distance value) and Vector Matching (which yields a cosine similarity score). The major contribution to achieve the given figure of matching accuracy can be deduced as the result of Tree Matching which uses the tree edit distance technique, since only a very small weightage (0.009) has been given to the cosine similarity score. Concerning the retrieval performance of the clustering algorithms, all of the four clustering algorithms have been able to achieve 100% training retrieval performance. The best testing retrieval performance has been achieved by the K-means and SOM+K-means clustering techniques, which averages around 96% and is better than the best results achieved by AHC and SOM. As a final point, the use of our proposed formula matching

technique together with the clustering techniques (in particular K-means or SOM+K-means) have demonstrated a feasible solution for a retrieval task that involves semantic-based retrieval of mathematical formulas.

CHAPTER 5

MATHEMATICAL DOCUMENT RETRIEVAL

In Chapter 4, we have discussed our approach for mathematical formula retrieval. This chapter discusses our proposed approach for retrieving mathematical documents, which can comprise formulas as well as text descriptions and graphical figures. As mentioned in Chapter 1, the purpose of retrieving mathematical documents is for problem solving. That is, for a given user's mathematical problem issued as a query, the goal is to retrieve a set of similar mathematical problems together with their associated solutions in order to provide hints or solutions on solving the user's problem. The proposed document retrieval approach investigates clustering techniques for the retrieval of similar mathematical documents. The clustering techniques explored here are similar to those explored for the formula search engine, that is, Agglomerative Hierarchical Clustering (AHC), K-means, Kohonen's Self-Organizing Map (SOM), and the hybrid clustering technique that integrates the SOM with K-means algorithm (SOM+K-means). In this chapter, the implementation and the performance evaluation of the proposed document retrieval approach will be discussed.

5.1 Proposed Approach

Figure 5-1 shows our proposed approach for mathematical document retrieval. In the proposed approach, a mathematical document database that stores a collection of mathematical documents is first constructed. As discussed in Chapter 3, each mathematical document contains two parts: a question part and a solution part. The question part and solution part may consist of any of the following three types of data, namely, plain text, mathematical formulas, and diagrams (e.g., geometrical figures). For instance, in the question part, plain text is used to describe the mathematical problem. For mathematical formulas, they are represented and stored in the mathematical document database using Mathematical Markup Language (MathML) [7] presentation markup similar to the storage of

formulas in the formula database as discussed in Chapter 4. For some documents, they can contain diagrams to illustrate a certain concept.

Figure 5-1: Proposed mathematical document retrieval approach.

The proposed approach can be explained in terms of two major processes: Training Process and Query Retrieval Process. The Training Process consists of the following four steps:

- Feature Selection;
- Preprocessing;

- Transformation; and
- Document Clustering.

Note that in the Training Process, we only train the question part of mathematical documents for knowledge generation. Once the Training Process completes, a knowledge model is generated and it is stored into the Document Knowledge Base for later use in query processing in the Query Retrieval Process. The Query Retrieval Process consists of four steps as follows:

- Query Feature Selection and Preprocessing;
- Query Transformation;
- Cluster Identification; and
- Document Ranking.

In the following sections, the steps involved in the Training Process and the Query Retrieval Process will be discussed in detail.

5.1.1 Feature Selection

The Feature Selection step aims to identify and extract the useful features from mathematical documents. The selected features should distinguish between different mathematical documents. By choosing only the essential features, it reduces the possibility of introducing noise into the overall process and also reduces the processing time of the system. Note that only the question part (or mathematical problem part) is considered for the extraction of useful features. The reason for that is because our ultimate goal is to retrieve a set of mathematical documents that consists of a similar question to a user's query problem.

From each mathematical document, two types of document content, namely, plain text descriptions and mathematical formulas are selected as representative features. On the other hand, diagrams in questions are usually stored as images which make feature extraction a complicated task. However, as illustrated in Figure 5-2 [100], diagrams are generally associated with text

descriptions (accompanied with mathematical formulas) for the corresponding explanation. Therefore, feature extraction from both plain text descriptions and mathematical formulas would be sufficient for our document retrieval purpose. Figure 5-3 shows an example of the Feature Selection step.

Figure 5-2: Example of question diagram described in the text description.

Figure 5-3: Example of Feature Selection step.

As shown in Figure 5-3, the Feature Selection step produces two types of document features: plain text descriptions and mathematical formulas. In the extracted plain text description, note that there is a tag titled “<eqn1>”. It represents the existence of the formula $\sqrt{7x^2 + 4}$ in the given position within the plain text description and the sequence number of the formula with respect to the whole document (i.e., here the sequence number is 1). Similarly, if there is a

diagram in a question, a tag titled “<diagX>” will be placed in the corresponding position within the plain text description, where X is the sequence number of the diagram.

5.1.2 Preprocessing

The Preprocessing step aims to remove noisy, erroneous, and insignificant data and prepare the representative features in a suitable format for knowledge generation. It consists of two sub-steps:

- Text Preprocessing; and
- Formula Preprocessing.

Text Preprocessing

Text Preprocessing preprocesses the descriptive plain text using text preprocessing techniques. It consists of four steps, namely, Noise Removal, Tokenization, Stemming, and Indicative Keyword Identification and Counting. These steps can be briefly described as follows:

- *Noise Removal.* It removes unwanted or excludable information from plain text descriptions, which are not meaningful for further processing (i.e., noise). These can include any of the following:
 - Words of length less than 3. e.g., p, q, at, of, an
 - Markup tags that represent mathematical formulas or diagrams. e.g., <eqn1>, <eqn2>, <diag1>
 - Numerical, roman, or alphabetical numberings. e.g., 1., (i), (ii), (a), (b)
 - Capital letter labels (that typically denote geometric figures or a set of points). e.g., AB, ABC, PQRS

- *Tokenization.* The Tokenization step divides the text string resulted from Noise Removal step into a sequence of word tokens. Since the target plain text descriptions are written in English, and since English words are separated by white spaces and punctuation marks, the tokenization algorithm makes use of white spaces and punctuation marks as separators between words. After tokenization, the resulting words are converted into lower case before they are passed into the next step, which is Stemming.
- *Stemming.* Stemming is the process of converting words into their root form. For example, when Stemming is applied to the three words “integrate”, “integral”, and “integration”, it will result in the same root word “integr”. In this step, the word tokens are converted into their root form. We used Porter’s stemming algorithm [82] for Stemming.
- *Indicative Keyword Identification and Counting.* This step identifies a set of indicative keywords from among the stemmed words. A list of indicative keywords is used for this purpose. The indicative keyword list, which comprises a total of 369 keywords (given in Appendix E), has been manually compiled according to heuristic features that can be typically observed in mathematical content. A snapshot of the list is shown in Figure 5-4. When the Indicative Keyword Identification and Counting step is carried out, stemmed words are matched with the entire keyword list. Note that at the time of matching, keywords in the indicative keyword list are also stemmed so that they can match the stemmed words. Each match will contribute to the corresponding (stemmed) indicative keyword count.

The use of indicative keyword list helps to extract the most significant terms that are capable of distinguishing different mathematical documents. In addition, it eliminates the need for a stop word removal process, which is commonly used to remove words with weak meaning, such as “the”, “and”, and “for”.

Indicative Keywords		
deceleration	definition	diagram
decimal	degree	difference
decrease	depth	differential
deduce	derivative	differentiate

Figure 5-4: Snapshot of the indicative keyword list.

Formula Preprocessing

In this step, mathematical formulas are preprocessed for the extraction of semantic meanings contained in the formulas. Since formulas are represented and stored in the system in the form of MathML presentation markup, this step extracts the semantic meanings of mathematical formulas from their MathML presentation markups. It consists of two sub-steps, namely, Regular Expression Matching and Keyword Stemming. These steps can be briefly described as follows:

- *Regular Expression Matching.* In this step, the semantic meanings of mathematical formulas are extracted as a series of indicative keywords (together with their frequency of occurrences). It is performed in a similar manner to the Formula Preprocessing procedure explained in Section 4.4.1. The same list of mathematical regular expressions (listed in Appendix A) is used in this case as well. The result is a set of indicative keywords with their frequency of occurrences obtained for each formula.
- *Keyword Stemming.* This step converts the indicative keywords collected during the Regular Expression Matching step into their root form. The idea here is to make sure that the (stemmed) indicative keywords obtained for the formulas are consistent with the (stemmed) indicative keywords obtained from Text Preprocessing. As such, Porter’s stemming algorithm [82] is used for stemming in this case as well.

After preprocessing plain text descriptions and mathematical formulas in each document, the resulting stemmed sets of keywords with their frequency of occurrences are counted and merged to form a keyword vector for each document. The resultant keyword vectors are then passed to the Transformation step for

further processing. The Preprocessing step is illustrated in Figure 5-5. It shows the steps in Preprocessing when applied on the result of the Feature Selection step given in Figure 5-3.

Figure 5-5: Preprocessing step example.

5.1.3 Transformation

The Transformation step applies a series of conversions to transform keyword vectors into normalized vectors. It is performed in a similar manner to the Transformation procedure explained in Section 4.4.1. As such, firstly, a TFIDF Conversion step is carried out, where the term frequencies in keyword vectors are re-weighted using the popular Term Frequency Inverse Document Frequency (TFIDF) weighting function [87]. Afterwards, a Normalization step normalizes the resulting TFIDF vectors into unit length using the Euclidean length of each vector.

When the Normalization step completes, a set of normalized vectors is generated, which are: (1) fed into the Document Clustering step for knowledge generation; and (2) stored into the Document Knowledge Base for later use in the Query Retrieval Process.

5.1.4 Document Clustering

In the Document Clustering step, the normalized vectors are fed into clustering algorithms for data clustering of similar mathematical documents. Similar to the clustering algorithms used for Formula Clustering as discussed in Section 4.4.2, the clustering algorithms used here include Agglomerative Hierarchical Clustering (AHC), K-means, Kohonen's Self-Organizing Map (SOM), and the hybrid clustering technique that integrates the SOM with the K-means algorithm (SOM+K-means). The selection of training parameters for the clustering algorithms are similar to our explanation given in Section 4.4.2, however with one exception for training the SOM (according to the first method of SOM training). The number of training samples in this case is 620, therefore the number of neurons (N) is given by: $N = 5 \times \sqrt{620} = 124.49 \approx 124$. This resulted a SOM grid with a dimensionality of 31×4 , which is rectangular in shape.

After setting-up the training parameters, the clustering algorithms have been trained using a similar procedure applied in the Formula Clustering step given in Section 4.4.2, using the normalized vectors fed as input. When the Document

Clustering step is completed, a set of knowledge models is generated by the clustering algorithms, which is stored in the Document Knowledge Base for subsequent use in query processing.

5.1.5 Document Knowledge Base

The Document Knowledge Base, similar to the Formula Knowledge Base discussed in Section 4.4.3, is a data storage that stores the different types of information generated during the Training Process. It basically consists of the following information:

- The set of normalized vectors generated at the completion of the Transformation step; and
- The set of document clusters (i.e., knowledge) generated as a result of training the AHC, K-means, SOM, and SOM+K-means clustering algorithms.

All of the information are stored in the Document Knowledge Base in the form of text files. Table 5-1 shows the details of the different types of information stored and the storage format of each information.

Table 5-1: Information stored in the Document Knowledge Base.

Information	Description	Storage Format
Normalized Vectors	The set of normalized vectors generated at the Transformation step. There is a single normalized vector for each document in the training data set.	Document_ID Dim_1 Dim_2 .. Dim_N
Cluster Centroid Vectors	The centroid vectors of the corresponding clusters generated by each clustering algorithm.	Cluster_ID Centriod_Dim_1 Centroid_Dim_2 .. Centroid_Dim_N
Documents in Each Cluster	For the clusters generated by each clustering algorithm, the documents contained in each cluster.	Document_ID,Cluster_ID
Meta-Information	Meta-information: number of documents and dimensionality of the normalized vectors.	Meta_info_1 Meta_info_2

5.1.6 Query Feature Selection and Preprocessing

When the Training Process is completed, the information stored in the Document Knowledge Base can then be used to find similar mathematical documents for a user's query during the Query Retrieval Process. The Query Retrieval Process begins with the Query Feature Selection and Preprocessing Step. In this step, the user's query is analyzed first to extract the representative features. As a result, two types of features: plain text descriptions and mathematical formulas, are extracted as the representative features, which is done in a similar manner to the Feature Selection step in the Training Process. Then, the selected features are preprocessed using similar techniques adopted in the Preprocessing step in the Training Process. The result is a query vector, which is eventually forwarded into the Query Transformation step for further processing.

5.1.7 Query Transformation

In this step, the generated query vector is transformed into a normalized query vector. It is performed in a similar manner to the query transformation process described in Section 4.4.4. The resulting normalized query vector is then passed into the Cluster Identification step for subsequent processing.

5.1.8 Cluster Identification

In the Cluster Identification step, the clusters containing the most similar mathematical documents to the user's query are determined. It is done in a similar manner to the Cluster Identification step discussed in Section 4.4.5. As a result, mathematical documents are retrieved from the mathematical document database starting from the nearest cluster, which are grouped according to the respective cluster of each mathematical document so that the proximity between each cluster (i.e., cluster centroid) and the user's query (i.e., normalized query vector) is also maintained.

5.1.9 Document Ranking

From the Cluster Identification step, sets of mathematical documents in the clusters have been retrieved, which are grouped according to the closeness of each cluster to the user's query. In this step, the retrieved mathematical documents are ranked and presented to the user as the query result. To do the ranking, each cluster is considered in turn according to its closeness to the query, and the normalized vectors corresponding to the documents in the cluster are retrieved from the Document Knowledge Base. Then, the cosine similarity is calculated between the normalized query vector and each normalized vector. This is similar to the Vector Matching process described in Section 4.4.7. As a result, a ranked set of mathematical documents is retrieved and displayed to the user.

5.2 Performance Evaluation

We have conducted experiments to evaluate the performance of our mathematical document retrieval approach. Basically, the training efficiency and the retrieval performance have been evaluated for the AHC, K-means, SOM, and SOM+K-means clustering algorithms. The performance measures are selected to be similar to those selected for the performance evaluation of the formula search engine as explained in Section 4.6. But, in this case, the relevance of a mathematical document has been assessed based on the entirety of features contained in the question part of the document (i.e., both plain text descriptions and mathematical formulas). In addition, the data set used for this evaluation is a mathematical document collection that consisted of 620 mathematical documents covering typical mathematical problems (and solutions) in the Differentiation and Integration topical areas in mathematics. Table 5-2 shows the composition of the document data set with some example mathematical documents (only the question part is shown) [94, 95, 96, 97, 99, 101, 112]. The entire document data set has been realized in terms of 8 different categories based on the semantic features of the mathematical problems.

Table 5-2: Composition of the mathematical document data set.

Document Category	Example Document (Question Part)	Feature Description
Limits	Find the limit of the following. $\frac{\sin 2x}{x}$ as $\lim_{x \rightarrow \infty}$.	Mathematical problems of finding limits.
Differentiation Techniques	Given that $y = \frac{(3x-2)^{10}}{6}$, find the value of $\frac{dy}{dx}$ when $x = \frac{1}{3}$.	Mathematical problems of solving a derivative.
Gradients, Tangents and Normals	The point $P(2,6)$ lies on the curve whose equation is $xy - y^2 + 12x = 0$. Find (i) the gradient of the curve at P , (ii) the angle which the tangent to the curve at P makes with the x-axis.	Mathematical problems involving finding gradients, tangents and normals.
Stationary, Maximum and Minimum Values/ Points	Two variables, x and y are related by the equation: $x + y = 3$. Find the maximum value of the variable z , where $z = 5x^3y$.	Mathematical problems of finding stationary, minimum, maximum values and/or points.
Approximations	Given that $y = x^3 + 9x$, use calculus to find, in terms of p , the approximate percentage increase in y when x increases from 3 to $3 + p$, where p is small.	Mathematical problems involving finding an approximation.
Integration Techniques	Evaluate the following definite integral. $\int_{-1}^1 (8x - 4) dx$	Mathematical problems of solving an integral.
Applications of Integration – Area and Volume	<p>The diagram shows part of the curve $y = 2 + \frac{1}{x+1}$. Find (i) the area of the shaded region, (ii) the volume generated when this region is rotated through 360° about the x-axis.</p>	Mathematical problems of applications of integration (i.e., using integration to calculate the area and volume).
Kinematics	A particle starts from a fixed point A and travels in a straight line. The velocity, $v \text{ ms}^{-1}$, of the particle, t seconds after leaving A is given by $v = 1 + t - \sqrt{4t + 9}$. Find the acceleration of the particle when it is at instantaneous rest.	Mathematical problems involving moving objects.

5.2.1 Training Efficiency

The training efficiency has been measured based on the time taken to complete one training session for AHC, K-means, SOM, and SOM+K-means clustering algorithms. The training efficiency has been measured for different training parameter configurations for each clustering algorithm.

Agglomerative Hierarchical Clustering Algorithm

Figure 5-6 shows the training efficiency of the Agglomerative Hierarchical Clustering (AHC) algorithm for different numbers of clusters (i.e., different stopping criterions). The horizontal axis is divided into equal-sized intervals of 5 for expected number of clusters ranging from 5 to 100. The vertical axis represents the total training time. The curve has been obtained by interpolating the total training time for different expected number of clusters. It can be seen that the total training time increases sub-linearly with the number of clusters.

Figure 5-6: Training efficiency of AHC.

K-means Algorithm

For the K-means algorithm, the total training time taken for different values of K is shown in Figure 5-7. The curve has been obtained by interpolating the total training time for different values of K . As shown in the figure, the total training time increases linearly with the number of clusters.

Figure 5-7: Training efficiency of K-means.

Kohonen's Self-Organizing Map Neural Network

Recall that the Kohonen's Self-Organizing Map (SOM) has been trained in two different methods. Figure 5-8 shows the total training time spent on different configurations of training iterations according to the first method of training the SOM (i.e., with fixed grid size (and shape) and varying numbers of training iterations). The curve has been obtained by interpolating the total training time for different numbers of training iterations. The total training time increases linearly with the number of training iterations.

Figure 5-8: Training efficiency for different numbers of training iterations configured on SOM.

Figure 5-9: Training efficiency for different numbers of neurons configured on SOM.

The training efficiency of SOM for different numbers of neurons configured on the grid (according to the second method of training the SOM) is shown in Figure 5-9. The curve has been obtained by interpolating the total

training time for different numbers of neurons (i.e., different numbers of clusters) configured on the SOM grid. It can be observed that the total training time increases as the number of neurons is increased.

SOM+K-means Hybrid Clustering Technique

Recall that the SOM+K-means clustering technique has been trained in two different methods depending on the SOM training: (1) with a fixed grid size (and shape) and varying numbers of training iterations; and (2) with a predetermined number of training iterations and varying numbers of neurons. In SOM+K-means, after training the SOM, K-means algorithm is trained using the output produced by SOM.

Figure 5-10: Training efficiency of SOM+K-means for different numbers of training iterations configured on SOM.

The total training time spent on training SOM+K-means for different numbers of training iterations configured on SOM (according to the first method of training the SOM) is shown in Figure 5-10. The curve has been obtained by interpolating the total training time for different numbers of training iterations

configured on SOM. It can be seen that the total training time increases linearly with the number of training iterations of SOM.

Figure 5-11 shows the training efficiency of SOM+K-means according to different numbers of neurons configured on SOM (according to the second method of SOM configuration). The curve has been obtained by interpolating the total training time for different numbers of neurons configured on SOM. The total training time increases linearly with the number of neurons configured on SOM.

Figure 5-11: Training efficiency of SOM+K-means for different numbers of neurons configured of SOM.

In summary, the training efficiency of AHC and K-means have been measured relative to the number of clusters generated. For SOM and SOM+K-means techniques, the total training time increased linearly with the number of neurons (i.e., number of clusters) and the number of training iterations configured on SOM. AHC and K-means have shown better training efficiency compared to SOM and SOM+K-means as the number of clusters is increased.

5.2.2 Retrieval Performance

Two types of retrieval performance evaluated here are the training retrieval performance and the testing retrieval performance for different clustering techniques. This evaluation has been conducted using a similar procedure as described in Section 4.5.2.

Training Retrieval Performance for Clustering Techniques

The goal of training retrieval performance evaluation is to measure the retrieval performance of the document search engine when the query problem already exists in the system’s document corpus. To conduct this evaluation, 30 test queries (listed in Appendix F) were designed by selecting a set of mathematical problems from the 8 different feature categories defined in the system. Each of the four clustering algorithms, AHC, K-means, SOM, and SOM+K-means, was trained according to the training procedure discussed in Section 5.1.4, and the average retrieval precision scores were obtained for each training session.

In summary, all of the four clustering techniques (AHC, K-means, SOM, and SOM+K-means) have achieved the best training retrieval performance of approximately 96%. The slight degradation in performance (i.e., not reaching 100%) is because, for certain test queries, the document containing the exact formula as in the query is not retrieved as the first ranked result. Nevertheless, the use of all of the four clustering techniques have been quite effective in retrieving an existing mathematical document in the system. Table 5-3 gives a summary of the best training retrieval performance for each clustering algorithm.

Table 5-3: Summary of best training retrieval performance for AHC, K-means, SOM, and SOM+K-means clustering techniques.

<i>Clustering Technique</i>	Best Training Retrieval Performance
AHC	96.11%
K-means	96.67%
SOM	96.67%
SOM+K-means	96.67%

Testing Retrieval Performance for Clustering Techniques

The testing retrieval performance has also been evaluated for the different clustering techniques. The goal here is to evaluate the retrieval performance of the document search engine when the query problem cannot be found in the training samples. However, it is similar to at least one or more mathematical problems in the training data set. To conduct this evaluation, 30 test queries (listed in Appendix F) have been designed. Each test query aims at retrieving one or more similar documents from one of the 8 feature categories defined in the system.

Figure 5-12: Testing retrieval performance for AHC and K-means.

The average retrieval precision scores obtained for the AHC and K-means algorithms are plotted in Figure 5-12. The horizontal axis denotes the number of clusters tested on a particular search session. Among the two algorithms, AHC achieved a better testing retrieval performance than K-means, which is around 86%. In addition, when considering the overall results, AHC achieved better results with larger numbers of clusters (e.g., greater than 60) than with smaller numbers of clusters.

Figure 5-13 shows the average retrieval precision values obtained for SOM and SOM+K-means clustering techniques relative to the number of training

iterations configured on SOM. SOM+K-means was able to achieve the best testing retrieval performance (about 88%) when the number of training iterations configured on SOM is 1000. In overall, both SOM and SOM+K-means have shown around 80%-86% testing retrieval performance.

Figure 5-13: Testing retrieval performance for SOM and SOM+K-means according to the number of training iterations configured on SOM.

Figure 5-14: Testing retrieval performance for SOM and SOM+K-means according to the number of neurons configured on SOM.

The testing retrieval performance for SOM and SOM+K-means according to the number of neurons configured on SOM is shown in Figure 5-14. On average, both techniques have demonstrated around 80-86% testing retrieval performance as the number of neurons was varied.

A summary of the best testing retrieval performance scores obtained for AHC, K-means, SOM, and SOM+K-means clustering techniques are given in Table 5-4. The table also shows the average online retrieval speed for the corresponding training parameter configuration. Among the four algorithms, the best testing retrieval performance has been achieved by the SOM+K-means technique.

Table 5-4: Summary of best testing retrieval performance for AHC, K-means, SOM, and SOM+K-means clustering techniques.

<i>Clustering Technique</i>	Training Parameter Configuration	Average Online Retrieval Speed	Best Testing Retrieval Performance
AHC	Number of Clusters = 90	1.55 sec	86.34%
K-means	Number of Clusters = 15	1.58 sec	84.77%
SOM	Number of Neurons = 124, Number of Iterations = 200	1.4 sec	86.18%
SOM+K-means	Number of Neurons (SOM) = 124, Number of Iterations (SOM) = 1000	1.45 sec	88.19%

In conclusion, the use of clustering techniques for semantic clustering of similar mathematical documents has been effective according to the performance scores revealed by the retrieval performance evaluation. The SOM+K-means hybrid clustering technique has discovered the similarity among mathematical documents more faithfully than the other clustering techniques.

5.3 Summary

In this chapter, we have presented our proposed approach for mathematical document retrieval, which ultimately aims at problem solving. For a user’s mathematical problem that is issued as a query, the system retrieves a set of similar

problems together with their associated solutions. The solutions associated with the retrieved problems can then be used to provide hints or solutions on solving the user's original query problem.

We have discussed the proposed approach in terms of two major processes: Training Process and Query Retrieval Process. In the Training Process, four clustering algorithms (AHC, K-means, SOM, and SOM+K-means) have been used to intelligently group similar mathematical documents and generate a Document Knowledge Base containing a set of knowledge models for the mathematical documents. In the Query Retrieval Process, the stored information in the Document Knowledge Base is used to find similar mathematical documents for a user's query problem.

The performance of our proposed document retrieval approach has been evaluated. Basically, the training efficiency and the retrieval performance have been evaluated for AHC, K-means, SOM, and SOM+K-means clustering techniques. It was observed that the best training retrieval performance achieved by all of the four clustering techniques is around 96%. The reason for not achieving 100% is because, for certain test queries, the document containing the exact formula as in the query was not retrieved as the first ranked result. The main reason behind this is that the regular expression-based matching technique, which works according to Vector Matching principle as discussed in Section 4.4.7, is sometimes unable to capture the exact features in a mathematical formula, causing the document containing the exact formula not to be ranked first. A possible solution for this would be to replace the matching technique with one that can better match the formula features. In terms of testing retrieval performance, it was observed that the hybrid clustering technique is able to achieve better results than the other clustering techniques. As such, it would be a feasible solution that can be used to develop a usable system in the future.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Summary

Mathematics is one of the most important subjects learned by students studying in schools and universities. A solid knowledge in mathematics is considered to be essential for a student's intellectual growth and logical thinking, which leads to a better understanding of many physical properties in the real world. Students can learn mathematics in different ways. One of the most effective ways is through solving mathematical problems. Nevertheless, solving mathematical problems is a challenging task for many students. To help students solve their mathematical problems, Computer Assisted Learning (CAL) tools have been gaining popularity, perhaps the most significant type of them are the computer algebra systems (CASs). However, the major issue in using CASs is that they can solve only certain types of mathematical problems, which are problems that can be issued in the form of symbolic notation (i.e., mathematical expression or formula). They cannot solve most of the mathematical problems that consist of a problem description, mathematical formulas, and diagrams. This research investigates the feasibility of using an information retrieval approach for mathematical problem solving, and experiments on the effectiveness of using data mining techniques (i.e., clustering) in the development of a search engine for mathematical document retrieval.

Our methodology for mathematical problem solving is that a user can issue his mathematical problem as a query into a mathematical document search engine. Then, the search engine retrieves a set of similar mathematical problems to the user's query problem together with their associated solutions. Subsequently, the retrieved problems and the solutions can be used by the user to get hints or solutions on solving his problem. In this project, a mathematical document is defined as a composite consisting of a mathematical problem and its corresponding solution.

After performing a detailed study on the features of mathematical documents, we observed that a significant proportion of semantics in the documents is formed by the mathematical formulas they contain. We also realized that there are several issues and concerns related to mathematical formulas and that the semantic matching between formulas is the most challenging issue in retrieving mathematical content. As such, we decomposed our overall goal of mathematical document retrieval into two sub-goals: formula search (i.e., finding a semantically similar mathematical formula) and document search (i.e., finding a similar mathematical document).

Concerning formula search, we conducted a detailed study to see what factors cause it difficult to retrieve mathematical formulas and the issues and considerations that are critical to the mathematical formula retrieval task. It was observed that although formulas are quite compact in their visual representation, they contain a lot of semantics in their small component parts built into their structure. After reviewing several existing formula search systems, we realized that the full-text retrieval techniques hardly suits formula search. Existing researches have tried different approaches for formula search, each of which has its own strengths and limitations.

To develop an approach for formula search, it was necessary to first decide the way in which mathematical formulas are represented in the system. We observed that using a standardized representation format would be beneficial than the other methods as we develop an information retrieval application. After analyzing the existing representation standards, we opted to make use of the Mathematical Markup Language (MathML) (specifically, its presentation markup) as an effective strategy to represent mathematical formulas in this work. After analyzing several existing projects, we could not find any system that makes use of data mining techniques (i.e., clustering) for mathematical formula retrieval. Since clustering can typically enhance the search results and has been widely adopted in several existing projects in other domains (e.g., bioinformatics), we decided to explore on using clustering techniques for intelligent retrieval of similar mathematical formulas. As mentioned before, the most crucial issue related to mathematical formula retrieval was observed as the semantic matching between

formulas. It is essential to use an effective formula matching technique that can address the various considerations related to formula search, so that most of the search needs of the users can be satisfied. After several experiments on finding a well-suited technique, we developed a new approach for formula matching, which is mainly based on the tree matching technique known as the *tree edit distance*, and also makes use of a mathematical regular expression list for feature extraction/identification from mathematical formulas.

A prototype formula search engine was implemented based on our proposed approach. For the developed system, we employed four different types of clustering techniques for the retrieval engine, namely, Agglomerative Hierarchical Clustering (AHC) algorithm which produces hierarchical clusters, K-means which performs partitional clustering, Kohonen's Self-Organizing Map (SOM) which is a neural network-based approach, and a hybrid clustering technique that integrates the SOM neural network with the K-means algorithm (SOM+K-means). An experiment was conducted to evaluate the performance of the formula search engine. The experimental results have shown that the proposed formula matching technique is able to achieve a very high matching accuracy. In addition, the system has shown very high retrieval accuracy in terms of training retrieval performance (i.e., finding an exact formula) and testing retrieval performance (i.e., finding a formula with similar features). In summary, our proposed approach has demonstrated the effectiveness of using clustering techniques to semantically group similar mathematical formulas and the effectiveness of the new matching technique for semantic matching between formulas.

Our next investigation was on mathematical document retrieval. Here it was necessary to derive an approach that can handle different types of information contained in mathematical documents (i.e., plain text descriptions, mathematical formulas, and diagrams). Since the ultimate goal of mathematical document retrieval is to retrieve a similar set of mathematical problems to a user's query problem, we only focused on the features contained in the question part of mathematical documents when developing our approach. In this case also (similar to formula search), we decided to investigate on using clustering techniques for semantic clustering and intelligent retrieval of similar mathematical documents.

To categorize the source documents into different clusters, it is necessary to extract the most representative features that can be used to distinguish different mathematical documents. We observed that whenever a diagram appears in the question part, it is typically described with some text description (accompanied by formulas) that gives an explanation about the diagram. As such, we decided that feature extraction from the plain text descriptions and formulas would be sufficient for our document retrieval purpose for this research. After conducting an analysis on the characteristics of typical mathematical problems, we observed that there are many indicative keywords in problem descriptions that give clear meanings to the underlying mathematical semantics. In other words, the terms used to express the problem in a typical mathematical problem description usually consisted of “mathematical terms” that provided direct meanings about what they state in the context of the problem. As such, we opted to use an indicative keyword list for the feature extraction from the problems when designing the retrieval engine. The same four clustering techniques, which were experimented for the formula search engine (i.e., AHC, K-means, SOM, and SOM+K-means) were experimented for the document retrieval engine as well.

A prototype mathematical document search engine was implemented based on our proposed approach, and experiments were conducted to evaluate its performance. The performance results indicated that our document retrieval approach can achieve good accuracy in retrieving similar mathematical documents based on a given mathematical problem. It signifies that our approach can be used as a feasible solution to help solve mathematical problems, in fact to solve virtually any possible mathematical problem.

In conclusion, this research has fully achieved its objectives. It has investigated many issues and concerns related to the retrieval of mathematical knowledge. The results indicate that our proposed approach for formula search can be used as a very good mechanism for searching similar mathematical formulas, and the mathematical document retrieval approach can be developed into a usable system as a feasible solution to help students studying in schools and universities in solving their mathematical problems.

6.2 Contributions

This research has made the following major contributions:

- Proposes an information retrieval approach for mathematical problem solving – This research, part of which was published in a conference [88], has explored on the feasibility of using an information retrieval approach for mathematical problem solving. It has particularly investigated on using clustering techniques for mathematical document retrieval, which has not yet been explored in any previous projects according to our knowledge. We have shown that our approach for mathematical document retrieval can achieve good results, and hence can be deployed into a usable system to help students in solving their mathematical problems.
- Introduces a new approach for formula search – In this research, we have proposed a new approach for mathematical formula matching, which is primarily based on the tree matching technique called *tree edit distance* accompanied by a regular expression-based formula feature extraction mechanism, and which has been investigated for the first time in the context of mathematical knowledge retrieval in this research to the best of our knowledge. We have shown that our formula matching approach can achieve very high level of accuracy, and it can handle most of the critical considerations related to mathematical formula search. In addition, we have investigated various clustering techniques for semantic clustering of similar mathematical formulas. We have shown that the use of clustering techniques together with our formula matching approach can achieve impressive results for mathematical formula search.

6.3 Future Work

The possible areas of future work on this research are summarized in the following sections.

6.3.1 Incorporating Features from Problem Solutions into the Clustering Process

A typical mathematical problem is always associated with one or more solutions. The features of the solutions may also help in distinguishing different types of mathematical problems in addition to the features of the problems themselves, and therefore solutions may also be incorporated into the clustering process in order to improve the accuracy of clustering. For instance, consider two mathematical problems related to Kinematics. One problem contains a formula giving a velocity, e.g., $v = 2 + pt - t^2$, and questions about an acceleration and the other problem contains the same velocity formula as the first problem and questions about a displacement. Among the two problems, the first problem is solved using differentiation techniques whereas the second problem is solved using integration techniques (which are the information available from the solutions of the two problems). Since the goal of mathematical document retrieval is to give a very closely matching problem that has a similar solution to the user's query problem, if the features of the solutions of the two problems can also be incorporated into the clustering process, it could further help in distinguishing the two mathematical problems more effectively.

6.3.2 Visualization of Search Results

It may be appealing to visualize the search results so that the user can get an intuitive understanding of the distribution of the search results with regard to his query. One possible way to do this would be to use the Kohonen's Self-Organizing Map (SOM) and text mining techniques to cluster the search results on-the-fly and represent it visually in a two-dimensional grid. Here, the keywords extracted from clusters may be used to describe the clusters and the relative distance between clusters may also be shown, so that the user will be able to understand the organization of his search results and also visualize clusters (i.e., search results) that are closer to the query. The cluster visualization may be used to browse the search results as well.

6.3.3 Incorporating an Improved Formula Search Mechanism for Mathematical Document Retrieval

The current mathematical document retrieval approach makes use of regular expressions for formula feature extraction (and matching). Although this approach has shown good accuracy, it is possible to further increase the level of accuracy. Some of the possible solutions can be given as follows:

- Combining search results of formula search and document search. In this research, we have proposed a formula search approach that can give very high accuracy in terms of formula matching as well as formula retrieval. As such, it would be possible to incorporate our formula search approach for the formulas in mathematical documents, and integrate the search results of formula search and mathematical document retrieval with the aim of better results for mathematical document retrieval.
- Replacing the existing formula matching approach with an entirely new approach. It is also possible to replace the existing formula matching approach in mathematical document retrieval (which is based on a mathematical regular expressions list) with an entirely new formula matching technique which can produce better results for mathematical document retrieval.

6.3.4 Conducting a Usability Study

The usefulness of the proposed mathematical document retrieval approach for mathematical problem solving would be best justified by conducting a user evaluation study. Due to the constraint on time, it was not possible to conduct such evaluation. As mentioned in our motivation statement, the typical users of this system would be students studying in schools and universities.

REFERENCES

- [1] M.E. Altamimi and A. Youssef, "Wildcards in Math Search, Implementation Issues," in *Proceedings of the ISCA 20th International Conference on Computer Applications in Industry and Engineering*, CAINE 2007, pp. 90-96, San Francisco, California, U.S.A., 2007.
- [2] T. Amagasa, L. Wen, and H.Kitagawa, "Proximity Search of XML Data Using Ontology and XPath Edit Similarity," in *Database and Expert Systems Applications*, Lecture Notes in Computer Science, Vol. 4653, pp. 298-307, Springer Berlin, 2007.
- [3] T. Aoki, K. Ota, K. Kurata, and T. Aoyagi, "Ordering Process of Self-Organizing Maps Improved by Asymmetric Neighborhood Function," in *Neural Information Processing*, Lecture Notes in Computer Science, Vol. 4984, pp. 426-435, Springer Berlin, 2008.
- [4] A. Asperti , F. Guidi, C. S. Coen, E. Tassi, and S. Zacchiroli, "A Content Based Mathematical Search Engine: Whelp," in *Types for Proofs and Programs*, Lecture Notes in Computer Science, Vol. 3839, pp. 17-32, 2006.
- [5] Association for the Advancement of Artificial Intelligence. AI Topics - Search. Available at:
<http://www.aaai.org/AITopics/pmwiki/pmwiki.php/AITopics/Search>.
- [6] A. Astel, S. Tsakovski, P. Barbieri, and V. Simeonov, "Comparison of self-organizing maps classification approach with cluster and principal components analysis for large environmental data sets," in *Water Research*, pp. 4566-4578, 2007.
- [7] R. Ausbrooks, S. Buswell, D. Carlisle, S. Dalmas, S. Devitt, A. Diaz, M. Froumentin, R. Hunter, P. Ion, M. Kohlhase, R. Miner, N. Poppelier, B. Smith, N. Soiffer, R. Sutor, and S. Watt, "Mathematical Markup Language (MathML) Version 2.0," World Wide Web Consortium (W3C) Recommendation, 21 February 2001. Available at:
<http://www.w3.org/TR/MathML2>.

References

-
- [8] G. Bancerek, "Information Retrieval and Rendering with MML Query," in *Mathematical Knowledge Management*, MKM 2006, Lecture Notes in Computer Science, Vol. 4108, pp. 266-279, Springer Verlag, 2006.
 - [9] P. Berkhin, "Survey of clustering data mining techniques," Technical Report, Accrue Software, 2002. Available at:
<http://citeseer.nj.nec.com/berkhin02survey.html>.
 - [10] P. Bille, "A Survey on the Tree Edit Distance and Related Problems," in *Theoretical Computer Science*, Vol. 337, No. 1-3, pp. 217-239, 2005.
 - [11] P.S. Bradley, U.M. Fayyad, and C. Reina, "Scaling Clustering Algorithm to Large Databases," in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pp. 9-15, New York, 1998.
 - [12] S. Buswell, O. Caprotti, D. Carlisle, M. Dewar, M. Gaetano, and M. Kohlhase: The OpenMath standard, version 2.0 (2004). Available at:
<http://www.openmath.org/>.
 - [13] Y. Cai and Q. Guo, "Point set generalization based on the Kohonen Net," in *Geo-Spatial Information Science*, Vol. 11, No. 3, pp. 221-227, 2008.
 - [14] P. Cairns, "Informalising Formal Mathematics: Searching the Mizar Library with Latent Semantics," in *Mathematical Knowledge Management*, MKM 2004, Lecture Notes in Computer Science, Vol. 3119, pp. 58-72, Springer Berlin, 2004.
 - [15] D.O. Case, *Looking for Information: A Survey of Research on Information Seeking Needs, and Behaviour*, California, USA: Elsevier Science, 2002.
 - [16] W.B. Chen, "An efficient hybrid clustering algorithm for molecular sequences classification," in *Proceedings of the 44th ACM Annual Southeast Regional Conference*, pp. 428-433, 2006.
 - [17] T.S. Chen, Y. T. Chen, C.C. Lin, and R. C. 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 Communications Systems*, pp. 405-408, 2005.
 - [18] Computing Relevance, Similarity: The Vector Space Model. Available at:
http://www.mtholyoke.edu/courses/dstrahma/cs341/myslides/Ch27b_ir2-vectorspace.ppt. Last Accessed Date: July 25, 2008.

References

- [19] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*, MIT Press, 2001.
 - [20] K. Daniels and C. Giraud-Carrier, "Learning the Threshold in Hierarchical Agglomerative Clustering," in *Proceedings of the 5th International Conference on Machine Learning and Applications*, ICMLA, pp. 270-278, 2006.
 - [21] Design Science, "Solutions for the Accessibility Community: What are the technology issues involved in making math accessible". Available at: <http://www.dessci.com/en/solutions/access/techissues.htm>.
 - [22] Design Science, "MathPlayer". Available at: <http://www.dessci.com/en/products/mathplayer>.
 - [23] Design Science, "WebEQ". Available at: <http://www.dessci.com/en/products/webeq>.
 - [24] M. C. Díaz-Galiano, M. Á. García-Cumbreras, M. T. Martín-Valdivia, A. Montejo-Ráez, and L. A. Ureña-López, "Integrating MeSH Ontology to Improve Medical Information Retrieval," in *Advances in Multilingual and Multimodal Information Retrieval*, Lecture Notes in Computer Science, Vol. 5152, pp. 601-606, Springer Berlin, 2008.
 - [25] S. Dulucq and L. Tichit, "RNA Secondary structure comparison: exact analysis of the Zhang--Shasha tree edit algorithm," in *Theoretical Computer Science*, Vol. 306, No. 1-3, pp. 471-484, 2003.
 - [26] S. Dulucq and H. Touzet, "Analysis of Tree Edit Distance Algorithms," in *Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching*, CPM 2003, Vol. 2676, pp. 1018, Morelia, Michoacán, Mexico, June 25-27, 2003.
 - [27] M.H. Dunham, *Data Mining: Introductory and Advanced Topics*, Pearson Education, Inc., 2003.
 - [28] B. Everitt, *Cluster Analysis*, 3rd Edition, London, England: Edward Arnold 1993.
 - [29] Google. Available at: <http://www.google.com/>.
 - [30] S. Grottke, S. Jeschke, N. Natho, and R. Seiler, "mArachna: A Classification Scheme for Semantic Retrieval in eLearning Environments in Mathematics," in *Proceedings of the 3rd International Conference on Multimedia and ICTs in Education*, June 7-10, Caceres/Spain, 2005.
-

References

- [31] M. Guegan and N. Hernandez, "Recognizing Textual Parallelisms with Edit Distance and Similarity Degree," in *Proceedings of EACL 2006*, The Association for Computer Linguistics, Trento, Italy, April 3-7, 2006.
 - [32] S.P. Harter, "Psychological Relevance and Information Science," in *Journal of American Society for Information Science*, Vol. 43, No. 9, pp. 602-615, 1992.
 - [33] J. Hartigan, *Clustering Algorithms*, New York, U.S.A., John Wiley & Sons, 1975.
 - [34] J. Hartigan and M. Wong, "Algorithm AS136: A K-means Clustering Algorithm," in *Applied Statistics*, Vol. 28, No. 1, pp. 100-108, 1979.
 - [35] S. Haykin, *Neural Networks and Learning Machines*, 3rd Edition, Prentice Hall, 2008.
 - [36] J. Hertz, A. Krogh, and R.G. Palmer, *Introduction to Theory of Neural Computation*, Addison-Wesley Longman Publication Corporation, 1991.
 - [37] C.M. Hoffmann and M.J. O'Donnell, "Pattern Matching in Trees," in *Journal of the ACM*, JACM, Vol. 29, No. 1, pp. 68-95, 1982.
 - [38] C.W. Holsapple, *Handbook of Knowledge Management: Knowledge Matters*, Vol. 2, pp. 168-171, 2007.
 - [39] N.-C. Hsieh, "Hybrid mining approach in the design of credit scoring models," in *Expert Systems with Applications*, Vol. 28, No. 4, pp. 655-665, 2005.
 - [40] M. Hussain and J.P. Eakins, "Component-based visual clustering using the self-organizing map," in *Neural Networks*, Vol. 20, No. 2, pp. 260-273, 2007.
 - [41] P. Ingwersen and P. Borlund, "Information Transfer Viewed as Interactive Cognitive Processes," in *Proceedings of the 2nd International Conference on Conceptions of Library Science: Integration in Perspective*, pp. 219-232, Copenhagen, Denmark, 1996.
 - [42] P. Jackson and I. Moulinier, *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization*, Second Revised Edition. John Benjamins Publishing Company, 2007.
 - [43] A. Jain, M. Murty, and P. Flynn, "Data clustering: A Review," in *ACM Computing Surveys*, Vol. 31, No. 3, pp. 264-233, 1999.
-

References

-
- [44] T. Jiang, L. Wang, and K. Zhang, "Alignment of Trees – An Alternative to Tree Edit," in *Combinatorial Pattern Matching, Lecture Notes in Computer Science*, Vol. 807, pp. 75-86, 1994.
 - [45] S.S. Kang, "Keyword-based Document Clustering," in *Proceedings of the 6th International Workshop on Information Retrieval with Asian Languages*, Sapporo, Japan, Vol. 11, pp. 132-137, 2003.
 - [46] D. Kauchak and S. Dasgupta, "An Iterative Improvement Procedure for Hierarchical Clustering," in *Advances in Neural Information Processing Systems*, Vol. 16, pp. 481-488, Great Britain, 2008.
 - [47] P. Kilpelainen and H. Mannila, "Ordered and Unordered Tree Inclusion," in *SIAM Journal on Computing*, Vol. 24, No. 2, pp. 340-356, 1995.
 - [48] B. King, "Step-wise clustering procedures," *Journal of Americal Statistics Association*, Vol. 62, No. 317, pp. 86-101, 1967.
 - [49] C.-C. Kiu and C.-S. Lee, "Ontological Knowledge Management Through Hybrid Unsupervised Clustering Techniques," in *Progress in WWW Research and Development, Lecture Notes in Computer Science*, Vol. 4976, pp. 499-510, Springer Berlin, 2008.
 - [50] P.N. Klein, "Computing the Edit-Distance Between Unrooted Ordered trees," in *Algorithms – ESA' 98, Lecture Notes in Computer Science*, Vol. 1461, pp. 1, Springer Berlin, 1998.
 - [51] H. Kruger, "Searching Mathematics with Zentralblatt Math: Overview and Outlook," in *Enhancing the Searching of Mathematics*, IMA "Hot Topics" Workshop, U.S.A., 2004.
 - [52] M. Kohlhase, "Markup for Mathematical Knowledge," in *OMDoc – An Open Markup format for Mathematical Documents*, Ver. 1.2, Lecture Notes in Computer Science, Vol. 4180, pp. 13-23, Springer Berlin, 2006.
 - [53] M. Kohlhase and I. Sucan, "A Search Engine for Mathematical Formulae," in *Proceedings of Artificial Intelligence and Symbolic Computation, AISC 2006*, Lecture Notes in Computer Science, Vol. 4120, pp 241-253, Springer Berlin, 2006.
 - [54] T. Kohonen, *Self-Organizing Maps*. Springer, 1995.
 - [55] T. Kohonen, *Self-Organizing Maps*, Berlin: Springer-Verlag, 2001.
-

References

- [56] Z. Kulpa, "On Diagrammatic Representation of Mathematical Knowledge," in *Proceedings of Mathematical Knowledge Management*, MKM 2004, Lecture Notes in Computer Science, Vol. 3119, pp. 190-204, Springer Berlin, 2004.
 - [57] M. Kun and B. Fuling, "An Ontology-Based Approach for Geographic Information Retrieval on the Web," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing*, WiCom 2007, pp. 5959-5962, 2007.
 - [58] R.J. Kuo, Y.L. An, H.S. Wang, and W.J. Chung, "Integration of self-organizing feature maps neural network and genetic K-means algorithm for market segmentation," in *International Journal of Expert Systems with Applications*, Vol. 30, No. 2, pp. 313-324, February 2006.
 - [59] R.J. Kuo, L.M. Ho, and C.M. Hu, "Integration of self-organizing feature map and K-means algorithm for market segmentation," in *Computers and Operations Research*, Vol. 29, No. 11, pp. 1475-1493, 2002.
 - [60] K.V. Laerhoven, "Combining the Self-Organizing Map and K-Means Clustering for On-Line Classification of Sensor Data," in *Artificial Neural Networks*, ICANN 2001, Lecture Notes in Computer Science, Vol. 2130, pp. 464-469, Springer Berlin, 2001.
 - [61] D.T. Larose, *Discovering Knowledge in Data: An Introduction to Data Mining*, pp. 16, John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.
 - [62] LaTeX project. Available at: <http://www.latex-project.org>.
 - [63] T.Y. Lim and E.K. Tang, "Effects of Cognitive and Problem Solving Style on Internet Search Tool," in *Research and Advanced Technology for Digital Libraries*, Lecture Notes in Computer Science, Vol. 1923, pp. 389-394, 2000.
 - [64] H.F. Ma, Q. He, and Z.Z. Shi, "Geodesic distance based approach for sentence similarity computation," in *Proceedings of the 2008 International Conference on Machine Learning and Cybernetics*, Vol. 5, pp. 2551-2557, 2008.
 - [65] C.D. Manning, P. Raghavan and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.
 - [66] Maple. Available at: <http://www.maplesoft.com/>.
 - [67] Mathematica. Available at:
<http://www.wolfram.com/products/mathematica/index.html>.
-

References

-
- [68] Mathematical Markup Language (MathML), Version 2.0 (Second Edition), Presentation Markup. Available at:
<http://www.w3.org/TR/MathML2/chapter3.html>.
 - [69] P. Mather and B. Tso, *Classification Methods for Remotely Sensed Data*, 2nd Edition, Routledge, U.S.A., 2001.
 - [70] MathWebSearch. Available at: <http://search.mathweb.org/index.xhtml>.
 - [71] C.T. Meadow, B.R. Boyce, and D.H. Kraft, *Text Information Retrieval Systems*, Library and Information Science, Second Edition, Academic Press, Inc., Orlando, FL, USA, 1999.
 - [72] T. Mikami and M. Wada, "Data Visualization Method for Growing Self-Organizing Networks with Ant Clustering Algorithm," in *Advances in Artificial Life*, Lecture Notes in Computer Science, Vol. 2159, pp. 623-626, Springer Berlin, 2001.
 - [73] R. Mikhael and E. Stroulia, "Examining Usage Protocols for Service Discovery," in *Service-Oriented Computing – ICSOC 2006*, Lecture Notes in Computer Science, Vol. 4294, pp. 496-502, Springer Berlin, 2006.
 - [74] B.R. Miller and A. Youssef, "Technical Aspects of the Digital Library of Mathematical Functions," in *Annals of Mathematics and Artificial Intelligence*, Vol. 38, No. 1-3, pp. 121-136, Springer Netherlands, 2003.
 - [75] R. Miner and R. Munavalli, "An Approach to Mathematical Search Through Query Formulation and Data Normalization," in *Towards Mechanized Mathematical Assistants*, Lecture Notes in Computer Science, Vol. 4573, pp. 342-355, Springer Berlin, 2007.
 - [76] J. Misutka and L. Galambos, "Mathematical Extension of Full Text Search Engine Indexer," in *Proceedings of the 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, ICTTA 2008, pp. 1-6, April 2008.
 - [77] M. Nanni, "Speeding-Up Hierarchical Agglomerative Clustering in Presence of Expensive Metrics," in *Advances in Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, Vol. 3518, pp. 378-387, Springer Berlin, 2005.
 - [78] NIST, "Digital Library of Mathematical Functions (DLMF)". Available at:
<http://dlmf.nist.gov>.
-

References

- [79] L. Padovani, "On the Roles of LATEX and MathML in Encoding and Processing Mathematical Expressions," in *Proceedings of the Second International Conference on Mathematical Knowledge Management*, MKM 2003, Vol. 2594, pp. 66-79, 2003.
 - [80] G.S. Pascal and W. Mineau, "Beyond TFIDF weighting for text categorization in the vector space model," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI, in Kaelbling, L.P. and Saffiotti, A. (Eds.): IJCAI-05, pp. 1130-1135, Professional Book Center 2005, ISBN 0938075934, Edinburgh, Scotland, UK.
 - [81] S.G. Piatetsky, "Data mining and knowledge discovery in business databases," in *Foundations of Intelligent Systems*, Lecture Notes in Artificial Intelligence, Vol. 1079, pp. 56-57, 1996.
 - [82] M.F. Porter, "An Algorithm for Suffix Stripping," *Program*, Vol. 14, No. 3, pp. 130-137, 1980.
 - [83] G. Punj and D.W. Steward, "Cluster analysis in marketing research: Review and suggestions for applications," in *Journal of Marketing Research*, Vol. 20, pp. 134-148, 1983.
 - [84] R. Ramesh and I.V. Ramakrishnan, "Nonlinear Pattern Matching in Trees," in *Journal of the ACM*, JACM, Vol. 39, No. 2, pp. 295-316, 1992.
 - [85] S. Revithis, W.H. Wilson, and N. Marcus, "IPSOM: A Self-organizing Map Spatial Model of How Humans Complete Interlocking Puzzles," in *AI 2006: Advances in Artificial Intelligence*, Lecture Notes in Computer Science, Vol. 4304, pp. 285-294, Springer Berlin, 2006.
 - [86] T. Sager, A. Bernstein, M. Pinzger, and C. Kiefer, "Detecting Similar Java Classes Using Tree Algorithms," in *Proceedings of the 2006 International Conference on Software Engineering - International workshop on Mining software repositories*, pp. 65-71, 2006.
 - [87] G. Salton, A. Wong, and C.S. Yang, "A Vector Space Model for Automatic Indexing," in *Communications of the ACM*, Vol. 18, No. 11, pp. 613-620, 1975.
 - [88] S. H. Samarasinghe and S.C. Hui, "Mathematical Document Retrieval for Problem Solving," in *Proceedings of the International Conference on Computer Engineering and Technology*, ICCET '08, pp. 583-587, 2009.
-

References

-
- [89] T. Saracevic, "Evaluation of Evaluation in Information Retrieval," in *Proceedings of the 18th ACM SIGIR Conference on Research and Development of Information Retrieval*, pp. 138-146, Seattle, 1995.
- [90] T. Saracevic, "Relevance Reconsidered. Information Science: *Integration in Perspective*," Copenhagen: Royal School of Librarianship, pp. 210-218, 1996.
- [91] M. Sasaki and H. Shinnou, "Spam Detection Using Text Clustering," in *Proceedings of the 2005 International Conference on Cyberworlds*, pp. 316-319, November 23-25, 2005.
- [92] M. Shatnawi and A. Youssef, "Equivalence Detection using Parse-Tree Normalization for Math Search," in *2nd International Conference on Digital Information Management*, ICDIM 2007, Vol. 2, pp. 643-648, 2007.
- [93] C. Silverstein, M. Henzinger, and H. Marais, "Analysis of a Very Large AltaVista Query Log," in *Digital SRC Technical Note*, #1998-014, October 26, 1998.
- [94] Singapore. Ministry of Education. *Singapore-Cambridge General Certificate of Education Ordinary Level Mathematics Examination Paper 2*. Singapore: Ministry of Education, June, 1996.
- [95] Singapore. Ministry of Education. *Singapore-Cambridge General Certificate of Education Ordinary Level Mathematics Examination Paper 2*. Singapore: Ministry of Education, June, 1997.
- [96] Singapore. Ministry of Education. *Singapore-Cambridge General Certificate of Education Ordinary Level Mathematics Examination Paper 1*. Singapore: Ministry of Education, November, 1997.
- [97] Singapore. Ministry of Education. *Singapore-Cambridge General Certificate of Education Ordinary Level Mathematics Examination Paper 1*. Singapore: Ministry of Education, November, 2000.
- [98] Singapore. Ministry of Education. *Singapore-Cambridge General Certificate of Education Ordinary Level Mathematics Examination Paper 2*. Singapore: Ministry of Education, November, 2000.
- [99] Singapore. Ministry of Education. *Singapore-Cambridge General Certificate of Education Ordinary Level Mathematics Examination Paper 1*. Singapore: Ministry of Education, June, 2001.
-

References

-
- [100] Singapore. Ministry of Education. *Singapore-Cambridge General Certificate of Education Ordinary Level Mathematics Examination Paper 2*. Singapore: Ministry of Education, June, 2001.
 - [101] Singapore. Ministry of Education. *Singapore-Cambridge General Certificate of Education Ordinary Level Mathematics Examination Paper 2*. Singapore: Ministry of Education, November, 2007.
 - [102] P. Sneath and R. Sokal, *Numerical Taxonomy*, London, England: Freeman, 1973.
 - [103] A. Stambuk, N. Stambuk, and P. Konjevoda, "Application of Kohonen's Self-Organizing Maps (SOM) Based Clustering for the Assessment of Religious Motivation," in *29th International Conference on Information Technology Interfaces*, pp. 87-91, 2007.
 - [104] T. Su, "Another look at non-random methods for initializing K-means clustering," Master Project, Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA, 2003.
 - [105] S. Sumathi and S.N. Sivanandam, *Introduction to Data Mining and Its Applications*, Springer Verlag, 2006.
 - [106] K.C. Tai, "The Tree-to-Tree Correction Problem," in *Journal of the ACM*, JACM, Vol. 26, No. 3, pp. 422-433, 1979.
 - [107] J. Tan, *E-Health Care Information Systems: An Introduction for Students and Professionals*, Jossey-Bass Inc., 2005.
 - [108] H.S. Tan and S. E. George, "Investigating Learning Parameters in a Standard 2-D SOM Model to Select Good Maps and Avoid Poor Ones," in *AI 2004: Advances in Artificial Intelligence*, Lecture Notes in Computer Science, Vol. 3339, pp. 425-437, Springer Berlin, 2004.
 - [109] P.N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Addison Wesley, Boston, 2006.
 - [110] The Wolfram Functions Site. Available at: <http://functions.wolfram.com>.
 - [111] G.B. Thomas and R.L. Finney, *Calculus and Analytic Geometry*, 9th Edition, Addison Wesley, 1995.
 - [112] G.B. Thomas, M.D. Weir, J. Hass, and F.R. Giordano, *Thomas' Calculus*, 9th Edition, Addison Wesley, 2004.
-

References

- [113] S. Ting and J. Dy, "A Deterministic Method for Initializing K-means Clustering," in *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI 2004, pp. 784-786, 2004.
 - [114] E. Tonisson. "Step-by-Step Solution Possibilities in Different Computer Algebra Systems," *ERIC*, EBSCOhost, 1999.
 - [115] A. Torsello and E.R. Hancock, "Tree Edit Distance from Information Theory," in *Graph Based Representations in Pattern Recognition*, Lecture Notes in Computer Science, Vol. 2726, pp. 705-759, Springer Berlin, 2003.
 - [116] A. Torsello, A. R.-Kelly, and E.R. Hancock, "Discovering Shape Classes using Tree Edit-Distance and Pairwise Clustering," in *International Journal of Computer Vision*, Vol. 72, No. 3, pp. 259-285, May 2007.
 - [117] H. Turtle and W.B. Croft, "Evaluation of an Inference Network-Based Retrieval Model," in *ACM Transactions on Information Systems*, Vol. 9, No. 3, pp. 187-222, 1991.
 - [118] C.J. Van Rijsbergen, *Information Retrieval*, 2nd Edition, Butterworths, London, England, 1979.
 - [119] J. Vesanto, "Neural network tool for data mining: SOM Toolbox," in *Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems*, TOOL-MET 2000, pp. 184-196, Oulun yliopistopaino, Oulu, Finland, 2000.
 - [120] Y. Wang, D.J. DeWitt, and J.Y. Cai, "X-Diff: an effective change detection algorithm for XML documents," in *Proceedings of the 19th International Conference on Data Engineering*, pp. 519-530, 2003.
 - [121] J. Ward, "Hierarchical grouping to optimize an objective function," in *Journal of American Statistics Association*, Vol. 58, pp. 236-244, 1963.
 - [122] S. Wesolkowski, R.D. Dony, and M.E. Jernigan, "Global color image segmentation strategies: Euclidean distance vs.vector angle," in *Proceedings of the 1999 IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing IX*, pp. 419-428, 1999.
 - [123] Wolfram, "Formula Search". Available at:
<http://functions.wolfram.com/formulasearch>.
 - [124] World Wide Web Consortium. Available at: <http://www.w3.org/>.
 - [125] World Wide Web Consortium (W3C), Document Object Model (DOM). Available at: <http://www.w3.org/DOM/>.
-

References

- [126] World Wide Web Consortium (W3C), Document Object Model for MathML. Available at: <http://www.w3.org/TR/MathML2/chapter8.html>.
- [127] Y. Xu, "Hybrid Clustering with Application to Web Mining," in *Proceedings of the 2005 International Conference on Active Media Technology*, AMT 2005, pp. 574-578, 2005.
- [128] Yahoo!. Available at: <http://www.yahoo.com/>.
- [129] R.B. Yates and B.R. Neto, *Modern Information Retrieval*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999.
- [130] L. Yin and C.-H. Huang, "Clustering of Gene Expression Data: Performance and Similarity Analysis," in *Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences*, IMSCCS '06, Vol. 1, pp. 142-149, 2006.
- [131] A. Youssef, "Roles of Math Search in Mathematics," in *Mathematical Knowledge Management*, MKM 2006, Lecture Notes in Computer Science, Vol. 4108, pp. 2-16, Springer Berlin, 2006.
- [132] A. Youssef, "Methods of Relevance Ranking and Hit-content Generation in Math Search," in *Towards Mechanized Mathematical Assistants*, Lecture Notes in Computer Science, Vol. 4573, pp. 393-406, 2007.
- [133] P. Zezula, G. Amato, V. Dohnal, and M. Batko, "Similarity Search: The Metric Space Approach," Springer, 2006.
- [134] H. Zhang, T.B. Ho, and M.S. Lin, "An Evolutionary K-means Algorithm for Clustering Time Series Data," in *Proceedings of International Conference on Machine Learning and Cybernetics*, Vol. 2, pp. 1282-1287, 2004.
- [135] J. Zhao, M.Y. Kan, and Y.L. Theng, "Math information retrieval: User Requirements and Prototype Implementation," in *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital Libraries*, Pittsburgh PA, PA, U.S.A., pp. 187-196, 2008.
- [136] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," in *SIAM Journal on Computing*, Vol. 18, No. 6, pp. 1245-1262, 1989.

Appendix A

List of
Mathematical Regular Expressions

Following is the list of 292 mathematical regular expressions that have been used to preprocess mathematical formulas.

Mathematical Regular Expression	Indicative Keyword
(?:&verbar | | <mo[^\>]*> </mo>).*?(?:&verbar | | <mo[^\>]*> </mo>)	absolute
(?:∠ ∠ ∠)	angle
<m(?:root)>\s*<mrow>.*?[a-ce-z].*?</mrow>\s*(?:<mn[^\>]*> $\sqrt{}$ </m(?:root)>	anyroot
(?:≈ ≈ ≈)	approximate
<mo[^\>]*>\s*≈</mo>	approximately
(<mi[^\>]*>A</mi>\s*<mo>=</mo>)	areaformula
(<mi[^\>]*>c</mi>\s*<mi[^\>]*>m</mi>)	centimeter
(?:δ δ δ)	change
<mi[^\>]*>\p{Alpha}</mi>\s*(?:<mtex[^\>]*> </mtex>\s*)*<mi[^\>]*>&ogr;</mi>\s*(?:<mtex[^\>]*> </mtex>\s*)*<mi[^\>]*>\p{Alpha}</mi>\s*	composite
<mi[^\>]*>[f]g{2}</mi>	composite
<mi[^\>]*>csc</mi>	cosec
<msup>\s*<mrow>\s*<mi[^\>]*>csc</mi>\s*</mrow>\s*<mrow>\s*<mo>−</mo>\s*<mn[^\>]*>1</mn>\s*</mrow>\s*</msup>	cosecinverse
(<msup>\s*<mrow>\s*<mi[^\>]*>csc{1}</mi>\s*(\s*<mo>⁡</mo>\s*)?</mrow>\s*<mrow>\s*<mn[^\>]*>8</mn>\s*</mrow>\s*</msup>)	cosecpowereight
(<msup>\s*<mrow>\s*<mi[^\>]*>csc{1}</mi>\s*(\s*<mo>⁡</mo>\s*)?</mrow>\s*<mrow>\s*<mn[^\>]*>5</mn>\s*</mrow>\s*</msup>)	cosecpowerfive
(<msup>\s*<mrow>\s*<mi[^\>]*>csc{1}</mi>\s*(\s*<mo>⁡</mo>\s*)?</mrow>\s*<mrow>\s*<mn[^\>]*>4</mn>\s*</mrow>\s*</msup>)	cosecpowerfour
(<msup>\s*<mrow>\s*<mi[^\>]*>csc{1}</mi>\s*(\s*<mo>⁡</mo>\s*)?</mrow>\s*<mrow>\s*<mn[^\>]*>9</mn>\s*</mrow>\s*</msup>)	cosecpowernine
(<msup>\s*<mrow>\s*<mi[^\>]*>csc{1}</mi>\s*(\s*<mo>⁡</mo>\s*)?</mrow>\s*<mrow>\s*<mn[^\>]*>7</mn>\s*</mrow>\s*</msup>)	cosecpowerseven
(<msup>\s*<mrow>\s*<mi[^\>]*>csc{1}</mi>\s*(\s*<mo>⁡</mo>\s*)?</mrow>\s*<mrow>\s*<mn[^\>]*>6</mn>\s*</mrow>\s*</msup>)	cosecpowersix
(<msup>\s*<mrow>\s*<mi[^\>]*>csc{1}</mi>\s*(\s*<mo>⁡</mo>\s*)?</mrow>\s*<mrow>\s*<mn[^\>]*>10</mn>\s*</mrow>\s*</msup>)	cosecpowerten
(<msup>\s*<mrow>\s*<mi[^\>]*>csc{1}</mi>\s*(\s*<mo>⁡</mo>\s*)?</mrow>\s*<mrow>\s*<mn[^\>]*>3</mn>\s*</mrow>\s*</msup>)	cosecpowerthree
(<msup>\s*<mrow>\s*<mi[^\>]*>csc{1}</mi>\s*(\s*<mo>⁡</mo>\s*)?</mrow>\s*<mrow>\s*<mn[^\>]*>2</mn>\s*</mrow>\s*</msup>)	cosecpowertwo
(<msup>\s*<mrow>\s*<mi[^\>]*>csc{1}</mi>\s*(\s*<mo>	cosecpowerwitheightx

Appendix A

List of Mathematical Regular Expressions

$\frac{\sec(x)}{\sec(x)} = \frac{1}{1} = 1$	
$\frac{\sec(x)}{\sec(x)} = \frac{1}{1} = 1$	cosecpowerwithfive
$\frac{\sec(x)}{\sec(x)} = \frac{1}{1} = 1$	cosecpowerwithfour
$\frac{\sec(x)}{\sec(x)} = \frac{1}{1} = 1$	cosecpowerwithnine
$\frac{\sec(x)}{\sec(x)} = \frac{1}{1} = 1$	cosecpowerwithseven
$\frac{\sec(x)}{\sec(x)} = \frac{1}{1} = 1$	cosecpowerwithsix
$\frac{\sec(x)}{\sec(x)} = \frac{1}{1} = 1$	cosecpowerwiththree

Appendix A

List of Mathematical Regular Expressions

$\frac{1}{\cos(x)}$	cosecpowerwithtwox
$\cos(8x)$	coseightx
$\cos(5x)$	cosfivex
$\cos(4x)$	cosfourx
$\cos(x)$	cosine
$\cos^2(x)$	cosineevenpower
$\cos^3(x)$	cosineoddpower

Appendix A

List of Mathematical Regular Expressions

$\frac{\cos(x^{\frac{1}{11}})^{\frac{1}{13}} \cos(x^{\frac{1}{15}})^{\frac{1}{17}} \cos(x^{\frac{1}{19}})^{\frac{1}{21}}}{\cos(x^{\frac{1}{8}})^{\frac{1}{5}} \cos(x^{\frac{1}{4}})^{\frac{1}{9}} \cos(x^{\frac{1}{7}})^{\frac{1}{6}} \cos(x^{\frac{1}{10}})^{\frac{1}{3}} \cos(x^{\frac{1}{2}})^{\frac{1}{2}}}$	
$\cos(x^{\frac{1}{8}})^{\frac{1}{5}}$	cosinepowereight
$\cos(x^{\frac{1}{5}})^{\frac{1}{4}}$	cosinepowerfive
$\cos(x^{\frac{1}{4}})^{\frac{1}{3}}$	cosinepowerfour
$\cos(x^{\frac{1}{9}})^{\frac{1}{7}}$	cosinepowernine
$\cos(x^{\frac{1}{7}})^{\frac{1}{6}}$	cosinepowerseven
$\cos(x^{\frac{1}{6}})^{\frac{1}{5}}$	cosinepowersix
$\cos(x^{\frac{1}{10}})^{\frac{1}{3}}$	cosinepowerten
$\cos(x^{\frac{1}{3}})^{\frac{1}{2}}$	cosinepowerthree
$\cos(x^{\frac{1}{2}})^{\frac{1}{2}}$	cosinepowertwo
$\cos(x^{\frac{1}{8}})^{\frac{1}{5}} \cos(x^{\frac{1}{4}})^{\frac{1}{9}} \cos(x^{\frac{1}{7}})^{\frac{1}{6}} \cos(x^{\frac{1}{10}})^{\frac{1}{3}} \cos(x^{\frac{1}{2}})^{\frac{1}{2}}$	cosinepowerwithheightx
$\cos(x^{\frac{1}{8}})^{\frac{1}{5}} \cos(x^{\frac{1}{4}})^{\frac{1}{9}} \cos(x^{\frac{1}{7}})^{\frac{1}{6}} \cos(x^{\frac{1}{10}})^{\frac{1}{3}} \cos(x^{\frac{1}{2}})^{\frac{1}{2}}$	cosinepowerwithfivex

Appendix A

List of Mathematical Regular Expressions

$\cos^{-1}(\rho, \sigma, \tau, \epsilon, \phi, \chi, \psi, \omega)$	
$\cos^{-1}\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	cosinepowerwithfourx
$\cos^{-1}\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	cosinepowerwithninx
$\cos^{-1}\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	cosinepowerwithsevenx
$\cos^{-1}\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	cosinepowerwithsixx
$\cos^{-1}\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	cosinepowerwiththreex
$\cos^{-1}\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	cosinepowerwithtwox
$\cos^{-1}\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	cosinverse

Appendix A

List of Mathematical Regular Expressions

$\cos^{-1}$	cosinverse
$\cos^{-1}(\sin(x))$	cosninx
$\cos(x)$	cosnx
$\cos(x) \cos(y)$	cosnxcosmxmultiplication
$\cos(\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta, \theta, \iota, \kappa, \lambda, \mu, \nu, \xi, \omicron, \pi, \rho, \sigma, \tau, \upsilon, \phi, \chi, \psi, \omega)$	
$\cos^7(x)$	cossevenx
$\cos^6(x)$	cossixx
$\cos^3(x)$	costhreeex
$\cos^2(x)$	costwox
$\cot$	cot
$\cot^{-1}$	cotinverse
$\cot^8(x)$	cotpowereight
$\cot^5(x)$	cotpowerfive
$\cot^4(x)$	cotpowerfour
$\cot^9(x)$	cotpowernine
$\cot^7(x)$	cotpowerseven
$\cot^6(x)$	cotpowersix
$\cot^{10}(x)$	cotpowerten
$\cot^3(x)$	cotpowerthree
$\cot^2(x)$	cotpowertwo

Appendix A

List of Mathematical Regular Expressions

$\cot^2 x$	cotpowerwitheightx
$\cot^5 x$	cotpowerwithfivex
$\cot^4 x$	cotpowerwithfourx
$\cot^9 x$	cotpowerwithninx
$\cot^7 x$	cotpowerwithsevenx
$\cot^6 x$	cotpowerwithsixx
$\cot^3 x$	cotpowerwiththreex

Appendix A

List of Mathematical Regular Expressions

$z]; \alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \}$	
$\cot^{\frac{1}{2}}$	cotpowerwithtwox
cm^3	cubiccentimeter
$\int$	definiteintegral
$\int_a^b$	definiteintegralnlowerlimit
$\int_a^b f(x) dx$	definiteintegralnupperlimit
$\int_a^b f(x) dx$	defintegralnonnumlowerlimit
$\int_a^b f(x) dx$	defintegralnonnumupperlimit
$\int_a^b f(x) dx$	defintegralnumlowerlimit
$\int_a^b f(x) dx$	defintegralnumupperlimit
$^\circ$	degree
$^\circ$	degrees
$\frac{d}{dx}$	differentiate
$\frac{d}{dx}$	differentiate
$\frac{d}{dx}$	differentiate
$'$	differentiate
$\frac{d}{dx}$	differentiate

Appendix A

List of Mathematical Regular Expressions

$\frac{d}{dx}$	differentiate
$\frac{d}{dx} \left(\frac{1}{x} \right)$	displacementformula
$\frac{d}{dx} \left(\frac{1}{x} \right)$	division
$\frac{d}{dx} \left(\frac{1}{x} \right)$	equalssign
$\frac{d}{dx} \left(\frac{1}{x} \right)$	equivalentto
e^{x^2}	exponential
e^{ix}	exponentialcomplexpower
$e^x - x^2$	exponentialminuspower
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpower
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpoweranyfraction
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowereight
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowerfive
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowerfour
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowerfourfifth
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowerhalf
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowernine
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpoweroneeighthth
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpoweronefifth
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpoweroneninth
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpoweroneseventh
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpoweronesixth
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpoweronethird
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowerquarter
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowerseven
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowersix
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowerten
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowerthree
$\frac{d}{dx} \left(\frac{1}{x} \right)$	expressionpowerthreefifth

Appendix A

List of Mathematical Regular Expressions

expressionpower2	expressionpower2
$\text{expressionpower2of5}$	$\text{expressionpower2of5}$
$\text{expressionpower2of3}$	$\text{expressionpower2of3}$
expressionpow3of4	expressionpow3of4
fifthderivative	fifthderivative
forthderivative	forthderivative
function	function
function	function
greaterthan	greaterthan
$\text{greaterthanorequalto}$	$\text{greaterthanorequalto}$
hasyterm	hasyterm
higherderivative	higherderivative

Appendix A

List of Mathematical Regular Expressions

>)	
<mi>csch</mi>	hyperboliccosec
<mi>cosh</mi>	hyperboliccosine
<mi>coth</mi>	hyperboliccot
<mi>sech</mi>	hyperbolicsec
<mi>sinh</mi>	hyperbolicsine
<mi>tanh</mi>	hyperbolictan
<mi>[[^]]*>∞</mi>	infinity
<mrow>\s*<mo>[[^]]*>∫</mo>	integral
<mo>[[^]]*>(?:> b > < ` < ≥ ≥ ≥ ≤ ≤ ≤ ≠ ≠ ≠)</mo>\s*<mi>[ [^] ]*>\p{Graph}+</mi>\s*<mo>[ [^] ]*>(?:> b > < ` < ≥ ≥ ≥ ≤ ≤ ≤ ≠ ≠ ≠)</mo>	interval
<mrow>.*?[\.]*?<mo>[[^]]*>.</mrow>.*?</mrow>	interval
<msup>\s*(?:<mi>[[^]]*>[fg]</mi>\s* .*?[\.]*?<mrow>\s*<mo>[[^]]*>(?:&minus ࢤ −)</mo>\s*<mn>[[^]]*>1</mn>\s*</mrow>\s*</msup>	inverse
((<mi>[[^]]*>sin</mi>\s*<mn>[[^]]*>-1</mn>(\s* .)* <mi>[[^]]*>sin</mi>\s*</mrow>\s*<mrow>\s*<mo>[[^]]*>*−</mo>(\s* .)* (<mi>[[^]]*>cos</mi>\s*<mn>[[^]]*>-1</mn>(\s* .)* <mi>[[^]]*>cos</mi>\s*</mrow>\s*<mrow>\s*<mo>[[^]]*>*−</mo>(\s* .)* (<mi>[[^]]*>tan</mi>\s*<mn>[[^]]*>-1</mn>(\s* .)* <mi>[[^]]*>tan</mi>\s*</mrow>\s*<mrow>\s*<mo>[[^]]*>*−</mo>(\s* .)*))	inversetrigonometric
(<mi>[[^]]*>k</mi>\s*<mi>[[^]]*>m</mi>)	kilometer
<mo>[[^]]*><</mo>	lessthan
<mo>[[^]]*>≤</mo>	lessthanorequalto
<munder>\s*<mrow>\s*<mi>[[^]]*>lim</mi>\s*</mrow>\s*<mrow>\s*<mi>[[^]]*>([a-z] α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ ϕ χ ψ ω){1}</mi>\s*<mo>→</mo>	limit
<munder>\s*<mrow>\s*<mi>[[^]]*>lim</mi>\s*</mrow>\s*<mrow>\s*<mi>[[^]]*>([a-z] α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ ϕ χ ψ ω){1}</mi>\s*<mo>→</mo>\s*<mi>[[^]]*>∞</mi>	limitreachinfinity
<munder>\s*<mrow>\s*<mi>[[^]]*>lim</mi>\s*</mrow>\s*<mrow>\s*<mi>[[^]]*>([a-z] α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ ϕ χ ψ ω){1}</mi>\s*<mo>→</mo>\s*<mn>0</mn>	limitreachzero
(<mi>[[^]]*>(ln log)</mi>\s*(<mo>⁡</mo>\s*)?(<mi>[[^]]*>(\(\)</mi>\s*?<mi>(ln log)</mi>)	lnInconsecutively
(<mi>[[^]]*>(?:ln log)</mi>)(<mi>[[^]]*>\s*\ln\s*</mi>\s*<mo>[[^]]*>\s*⁡\s*</mo>)	logarithm
(<mi>[[^]]*>(?:ln log)</mi>\s*<mo>(\(\)</mo>)(<mi>[[^]]*>\s*\ln\s*</mi>\s*<mo>[[^]]*>\s*⁡\s*</mo>\s*<mo>(\(\)</mo>)	logarithmexpression
(?:&map ↦ ↦)	map
</msup>\s*<mo>\(</mo>)(</msup>\s*<msqrt>)(</mi>\	multiplication

Appendix A

List of Mathematical Regular Expressions

[illegible]

Appendix A

List of Mathematical Regular Expressions

$\frac{m^7}{n^7}$	
$\frac{m^9}{n^9}$	polynomialpowerseven
$\frac{m^6}{n^6}$	polynomialpowersix
$\frac{m^{10}}{n^{10}}$	polynomialpowerten
$\frac{m^3}{n^3}$	polynomialpowerthree
$\frac{m^3}{n^5}$	polynomialpowerthreefifth
$\frac{m^2}{n^2}$	polynomialpowertwo
$\frac{m^2}{n^5}$	polynomialpowertwofifth
$\frac{m^1}{n^8}$	polynomialpowoneeightth
$\frac{m^1}{n^9}$	polynomialpowonenineth
$\frac{m^1}{n^7}$	polynomialpowoneseventh
$\frac{m^1}{n^6}$	polynomialpowonesixth
$\frac{m^1}{n^3}$	polynomialpowonethird
$\frac{m^3}{n^4}$	polynomialpowthreequarter

Appendix A

List of Mathematical Regular Expressions

$mn \cdot \frac{1}{m} \cdot \frac{1}{n}$	
$(x^2 + 1)^3$	polynomialpowtwothird
$\frac{d}{dx} \left(\frac{1}{x^2} \right)$	rate
$\frac{d}{dx} \left(\frac{1}{x^2} \right)$	reals
$\sqrt{x^2 + 1}$	root
$\sec(x)$	sec
$\frac{1}{\sec(x)}$	secinverse
$\frac{d^2}{dx^2} \left(\frac{1}{x^2} \right)$	secondderivative
$\sec^8(x)$	secpowereight
$\sec^5(x)$	secpowerfive
$\sec^4(x)$	secpowerfour
$\sec^9(x)$	secpowernine
$\sec^7(x)$	secpowerseven
$\sec^6(x)$	secpowersix
$\sec^{10}(x)$	secpowerten
$\sec^3(x)$	secpowerthree
$\sec^2(x)$	secpowertwo
$\sec^{100}(x)$	secpowerwitheightx

Appendix A

List of Mathematical Regular Expressions

$\frac{\sec(x)}{x^8} = \frac{1}{x^8} \cdot \frac{1}{\cos(x)}$	
$\frac{\sec(x)}{x^5} = \frac{1}{x^5} \cdot \frac{1}{\cos(x)}$	secpowerwithfivex
$\frac{\sec(x)}{x^4} = \frac{1}{x^4} \cdot \frac{1}{\cos(x)}$	secpowerwithfourx
$\frac{\sec(x)}{x^9} = \frac{1}{x^9} \cdot \frac{1}{\cos(x)}$	secpowerwithninx
$\frac{\sec(x)}{x^7} = \frac{1}{x^7} \cdot \frac{1}{\cos(x)}$	secpowerwithsevenx
$\frac{\sec(x)}{x^6} = \frac{1}{x^6} \cdot \frac{1}{\cos(x)}$	secpowerwithsixx
$\frac{\sec(x)}{x^3} = \frac{1}{x^3} \cdot \frac{1}{\cos(x)}$	secpowerwiththreex

Appendix A

List of Mathematical Regular Expressions

$\frac{1}{\sec^2(\alpha + \beta + \gamma + \delta + \epsilon + \zeta + \eta + \theta + \iota + \kappa + \lambda + \mu + \nu + \xi + \omicron + \pi + \rho + \sigma + \tau + \upsilon + \phi + \chi + \psi + \omega)}$	secpowerwithtwox
$\sin$	sin
$\frac{1}{\sin^2(\alpha + \beta + \gamma + \delta + \epsilon + \zeta + \eta + \theta + \iota + \kappa + \lambda + \mu + \nu + \xi + \omicron + \pi + \rho + \sigma + \tau + \upsilon + \phi + \chi + \psi + \omega)}$	sineevenpower
$\frac{1}{\sin^8(\alpha + \beta + \gamma + \delta + \epsilon + \zeta + \eta + \theta + \iota + \kappa + \lambda + \mu + \nu + \xi + \omicron + \pi + \rho + \sigma + \tau + \upsilon + \phi + \chi + \psi + \omega)}$	sineeightx
$\sin(\alpha + \beta + \gamma + \delta + \epsilon + \zeta + \eta + \theta + \iota + \kappa + \lambda + \mu + \nu + \xi + \omicron + \pi + \rho + \sigma + \tau + \upsilon + \phi + \chi + \psi + \omega) \cos(\alpha + \beta + \gamma + \delta + \epsilon + \zeta + \eta + \theta + \iota + \kappa + \lambda + \mu + \nu + \xi + \omicron + \pi + \rho + \sigma + \tau + \upsilon + \phi + \chi + \psi + \omega)$	sinenxcosmxmultiplication
$\sin(\alpha + \beta + \gamma + \delta + \epsilon + \zeta + \eta + \theta + \iota + \kappa + \lambda + \mu + \nu + \xi + \omicron + \pi + \rho + \sigma + \tau + \upsilon + \phi + \chi + \psi + \omega) \sin(\alpha + \beta + \gamma + \delta + \epsilon + \zeta + \eta + \theta + \iota + \kappa + \lambda + \mu + \nu + \xi + \omicron + \pi + \rho + \sigma + \tau + \upsilon + \phi + \chi + \psi + \omega)$	sinenxsinemxmultiplication
$\frac{1}{\sin^2(\alpha + \beta + \gamma + \delta + \epsilon + \zeta + \eta + \theta + \iota + \kappa + \lambda + \mu + \nu + \xi + \omicron + \pi + \rho + \sigma + \tau + \upsilon + \phi + \chi + \psi + \omega)}$	sineoddpower

Appendix A

List of Mathematical Regular Expressions

[illegible]

Appendix A

List of Mathematical Regular Expressions

$\theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \}$	
$\sin^5\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	sinepowerwithfive
$\sin^4\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	sinepowerwithfour
$\sin^9\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	sinepowerwithnine
$\sin^7\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	sinepowerwithseven
$\sin^6\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	sinepowerwithsix
$\sin^3\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	sinepowerwiththree
$\sin^2\left(\frac{a-cz}{\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega}\right)$	sinepowerwithtwo

Appendix A

List of Mathematical Regular Expressions

$\frac{a-cz}{1}$ α β γ δ ϵ ζ η θ ι κ λ μ ν ξ $\omicron$ π ρ σ τ υ ϕ χ ψ ω	
$\sin^2 x$	sinetwox
$\sin^5 x$	sinfivex
$\sin^4 x$	sinfourx
$\sin^{-1} x$	sininverse
$\arcsin x$	sininverse
$\sin^9 x$	sinninex
$\sin^n x$	sinnx
$\sin^7 x$	sinsevenx
$\sin^6 x$	sinsixx
$\sin^3 x$	sinthreex
cm^2	squarecentimeter
km^2	squarekilometer
$\sqrt{\quad}$	squareroot
$\tan$	tan
$\tan^8 x$	taneightx
$\tan^{2n} x$	tanevenpower

Appendix A

List of Mathematical Regular Expressions

[illegible]

Appendix A

List of Mathematical Regular Expressions

[illegible]

Appendix A

List of Mathematical Regular Expressions

$\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}}$	
$\tan^7 \left(\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}} \right)$	tanpowerwithsevenx
$\tan^6 \left(\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}} \right)$	tanpowerwithsixx
$\tan^3 \left(\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}} \right)$	tanpowerwiththreex
$\tan^2 \left(\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}} \right)$	tanpowerwithtwox
$\tan^7 \left(\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}} \right)$	tansevenx
$\tan^6 \left(\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}} \right)$	tansixx
$\tan^3 \left(\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}} \right)$	tanthreex
$\tan^2 \left(\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}} \right)$	tantwox
$\lim_{x \rightarrow \frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}}} \left(\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}} \right)$	tend
$\lim_{x \rightarrow \frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}}} \left(\frac{a - ce - z}{\alpha; \beta; \gamma; \delta; \epsilon; \zeta; \eta; \theta; \iota; \kappa; \lambda; \mu; \nu; \xi; \omicron; \pi; \rho; \sigma; \tau; \upsilon; \phi; \chi; \psi; \omega; \{1\}} \right)$	tendtoinfinity

Appendix A

List of Mathematical Regular Expressions

>\s*	
<mi ^{[^>]*} >[a-z]</mi>\s*<mo ^{[^>]*} >→</mo>\s*<mo ^{[^>]*} >−</mo>(\s*\.)*	tendtonegativevalue
<mi ^{[^>]*} >[a-z]</mi>\s*<mo ^{[^>]*} >→</mo>\s*<mn ^{[^>]*} >0</mn>\s*	tendtozero
(?:<mfrac>\s*<mrow>\s*<msup>\s*<mrow>\s*<mi>d</mi>\s*</mrow>\s*</mrow>\s*<mn>3</mn>\s*</mrow>\s*</msup>\s*<mi>y a-z</mi>\s*</mrow>\s*<mrow>\s*<mi>d</mi>\s*<msup>\s*<mrow>\s*<mi>x a-z</mi>\s*</mrow>\s*<mrow>\s*<mn>3</mn>\s*</mrow>\s*</msup>\s*</mrow>\s*</mfrac>) (?:<mfrac>\s*<mrow>\s*<msup>\s*<mrow>\s*<mo>ⅆ</mo>\s*</mrow>\s*<mrow>\s*<mn>3</mn>\s*</mrow>\s*</msup>\s*<mi>y a-z</mi>\s*</mrow>\s*<mrow>\s*<mo>ⅆ</mo>\s*<msup>\s*<mrow>\s*<mi>x a-z</mi>\s*</mrow>\s*<mrow>\s*<mn>3</mn>\s*</mrow>\s*</msup>\s*</mrow>\s*</mfrac>)	thirdderivative
<mi ^{[^>]*} >sin cos tan csc sec cot</mi>	trigonometric
<mi ^{[^>]*} >\s*(sin cos tan csc sec cot)\s*</mi>\s*<mo ^{[^>]*} >\s*⁡\s*</mo>\s*((\s*<mo>\s*</mo>\s*) (\s*<mo>\s*</mo>\s*))\s*<mn>[\s*\.]*\s*	trigononx
(<msup>\s*<mrow>\s*<mi ^{[^>]*} >(sin cos tan csc sec cot){1}</mi>\s*(\s*<mo>⁡</mo>\s*)?</mrow>\s*<mrow>\s*<mfrac>\s*<mrow>\s*)?((<mn ^{[^>]*} >(d)+</mn>) <mi>[a-ce-z]{1}</mi>))	trigonopower
(?:> > > < < < ≥ ≥ ≥ ≤ ≤ ≤ ≠ ≠ ≠ ≈)	unequal
<mi ^{[^>]*} >v</mi>\s*<mo>=</mo>	velocityformula
<mi ^{[^>]*} >V</mi>\s*<mo>=</mo>	volumeformula

Appendix B

List of Composite MathML Elements

The list of Composite MathML Elements that have been used in the process of constructing the Expression Tree for a mathematical formula is given below.

1. `<mfrac>`
2. `<msqrt>`
3. `<mroot>`
4. `<mstyle>`
5. `<merror>`
6. `<mpadded>`
7. `<mphantom>`
8. `<mfenced>`
9. `<menclose>`
10. `<msub>`
11. `<msup>`
12. `<msubsup>`
13. `<munder>`
14. `<mover>`
15. `<munderover>`
16. `<mmultiscripts>`
17. `<mtable>`
18. `<mlabeledtr>`
19. `<mtr>`
20. `<mtd>`
21. `<maction>`

Appendix C

Formula Data Set for Matching Accuracy Evaluation

Following is the sample formula data set that has been used to evaluate the matching accuracy of the formula matching technique. It is organized into 6 clusters based on the features of the mathematical formulas.

Cluster - Polynomial Functions

1. $\int (x + 1) \, dx$
2. $\int (x + 1)^2 \, dx$
3. $\int (2x + 1) \, dx$
4. $\int (x^2 + 1) \, dx$
5. $\int (x + 1)(x + 2) \, dx$
6. $\int (x + 1)^2 (x + 2) \, dx$
7. $\int (x^2 + 1)(x + 2) \, dx$
8. $\int (x + 1)(x + 2)(x + 3) \, dx$
9. $\int (x + 1)(x + 2)(x + 3)(x + 4) \, dx$
10. $\int (x + 2)(x + 1) \, dx$
11. $\int (t + 1)(t + 3)(t + 2) \, dt$
12. $\int (x^2 + x + 1) \, dx$
13. $\int (x^3 + x^2 + x + 1) \, dx$
14. $\int (x^4 + x^3 + x^2 + x + 1) \, dx$
15. $\int (x^5 + x^4 + x^3 + x^2 + x + 1) \, dx$
16. $\int (x^4 - x^3 + x^2 - x + 1) \, dx$
17. $\int (x^5 - x^4 + x^3 - x^2 + x - 1) \, dx$
18. $\int (x^5 - x^4 - x^3 - x^2 - x - 1) \, dx$
19. $\int (x^2 + 1)^2 \, dx$
20. $\int (x^2 + 1)^3 \, dx$

Cluster - Rational Functions

1. $\int \frac{dx}{x^2}$

2. $\int \frac{dx}{x^3}$

3. $\int \frac{dx}{x^2 + 1}$

4. $\int \frac{dx}{x^2 + x + 1}$

5. $\int \frac{1}{x^3 + x^2 + x + 1} dx$

6. $\int \frac{u + 1}{u^2 + u + 1} du$

7. $\int \frac{x^2 + x + 1}{x + 1} dx$

8. $\int \frac{dx}{(x + 1)(x + 2)}$

9. $\int \frac{dx}{(x + 1)(x + 2)(x + 3)}$

10. $\int \frac{dx}{(x + 1)^2(x + 2)}$

11. $\int \frac{x + 1}{x + 2} dx$

12. $\int \frac{x + 1}{(x + 2)(x + 3)} dx$

13. $\int \frac{x^3 + 1}{(x + 2)(x + 3)} dx$

14. $\int \frac{x^3 + 1}{(x + 1)(x + 2)(x + 3)} dx$

15. $\int \frac{x^3 + 1}{(x + 1)^2(x + 2)(x + 3)} dx$

16. $\int \frac{1}{(x + 1)^2} dx$

17. $\int \frac{x - 1}{(x + 1)^2} dx$

$$18. \int \left(x^2 + \frac{1}{x^2} \right) dx$$

$$19. \int \left(x + \frac{1}{x} \right) dx$$

$$20. \int \left((x+1)^3 + \frac{1}{(x+1)^3} \right) dx$$

Cluster - Irrational Functions

$$1. \int \sqrt{x} \, dx$$

$$2. \int \sqrt{x+1} \, dx$$

$$3. \int \sqrt{\sqrt{x}+1} \, dx$$

$$4. \int \sqrt{x^3+1} \, dx$$

$$5. \int \sqrt{1-u^2} \, du$$

$$6. \int x\sqrt{1-x^2} \, dx$$

$$7. \int x^3\sqrt{1-x^2} \, dx$$

$$8. \int \frac{1}{\sqrt{1-x^2}} \, dx$$

$$9. \int \frac{x}{\sqrt{1-x^2}} \, dx$$

$$10. \int \frac{\sqrt{x}+1}{\sqrt{x^2+3x+2}} \, dx$$

$$11. \int \frac{1}{\sqrt{x^2+3x+4}} \, dx$$

$$12. \int \frac{1}{\sqrt{(x+1)(x+2)}} \, dx$$

$$13. \int \frac{1}{\sqrt{(x+1)(x+2)(x+3)}} \, dx$$

$$14. \int \frac{1}{(x+2)\sqrt{(x+1)}} \, dx$$

$$15. \int \frac{1}{(x+2)^2\sqrt{(x+1)}} \, dx$$

$$16. \int \frac{1}{(\sqrt{x} + 2)^2 (\sqrt{x} + 1)} dx$$

$$17. \int \left(\sqrt{x} + \frac{1}{\sqrt{x}} \right) dx$$

$$18. \int (\sqrt{x} + 1)^2 (\sqrt{x} + 2) dx$$

$$19. \int \frac{1}{(1+x)^{\frac{3}{2}}} dx$$

$$20. \int \frac{1}{(1+x^2)^{\frac{3}{2}}} dx$$

Cluster - Trigonometric Functions

$$1. \int \sin x \, dx$$

$$2. \int \sin(2x) \, dx$$

$$3. \int \sin(2x + a) \, dx$$

$$4. \int \sin(x^2) \, dx$$

$$5. \int \sin \sqrt{x} \, dx$$

$$6. \int \arcsin x \, dx$$

$$7. \int \sin x \sin 2x \sin 3x \, dx$$

$$8. \int \sin^2 x \, dx$$

$$9. \int \cos x \, dx$$

$$10. \int \cos 3x \, dx$$

$$11. \int \cos(ax + b) \, dx$$

$$12. \int \cos x \cos(2x) \, dx$$

$$13. \int \cos x \cos 2x \cos 3x \, dx$$

$$14. \int \cos^4 x \, dx$$

$$15. \int \sin x \cos x \, dx$$

$$16. \int \sin x \cos x \cos x^2 \, dx$$

$$17. \int \sin x \tan x \, dx$$

$$18. \int \sin x \tan x \sin 2x \, dx$$

Appendix CFormula Data Set for Matching Accuracy Evaluation

19. $\int \frac{\sin x + \cos x}{\tan x} dx$
20. $\int (\sin x + \cos x) \tan x dx$
21. $\int \tan x dx$
22. $\int \tan^4 x dx$
23. $\int \sin x \cos x \tan x dx$
24. $\int x \sin x dx$
25. $\int x^2 \sin x dx$
26. $\int x \cos x dx$
27. $\int (x^2 + x + 1) \sin x \cos x dx$
28. $\int (x^2 + 1) \sin x dx$
29. $\int (\sin x + \cos x) dx$
30. $\int (3 \sin 4x + 4 \cos 3x) dx$

Cluster - Logarithmic and Exponential Functions

1. $\int e^x dx$
2. $\int e^{2x+1} dx$
3. $\int x e^x dx$
4. $\int x^2 e^x dx$
5. $\int t \ln t dt$
6. $\int x^2 \ln x dx$
7. $\int \frac{\ln x}{x^2} dx$
8. $\int \ln(x^2) dx$
9. $\int x (\ln x)^2 dx$
10. $\int \frac{e^x + 1}{e^x - 1} dx$
11. $\int \frac{dx}{e^x + 1}$
12. $\int \ln x \ln(2x) dx$

13. $\int \ln x \, dx$

14. $\int \frac{dx}{\ln x}$

15. $\int \ln\left(\frac{1}{x}\right) dx$

16. $\int \ln \ln x \, dx$

17. $\int x \ln \ln x^2 \, dx$

18. $\int \frac{\ln \ln x}{x} \, dx$

19. $\int \frac{\ln \ln x}{x^2} \, dx$

20. $\int x^2 \ln \ln x \, dx$

Cluster - Mixed Features

1. $\int e^x \sin x \, dx$

2. $\int e^x \sin x \cos x \, dx$

3. $\int e^x \ln x \sin x \, dx$

4. $\int \ln x \sin x \, dx$

5. $\int \ln x \tan x \, dx$

6. $\int \ln x \cos x \, dx$

7. $\int \ln x \sin x \cos x \, dx$

8. $\int x \ln x \sin x \cos x \, dx$

9. $\int \ln x \sin(\ln x) \, dx$

10. $\int \ln x \cos(\ln x) \, dx$

11. $\int \sin x \ln(\cos x) \, dx$

12. $\int \cos x \ln(\sin x) \, dx$

13. $\int \tan x \ln(\sin x) \, dx$

14. $\int \ln(\sin x \cos x) \, dx$

15. $\int \cos x \ln(\sin x \cos x) \, dx$

16. $\int \cos 2x \ln(\sin x \cos x) \, dx$

17. $\int e^x \tan x \, dx$

18. $\int e^{\sin x} \cos x \, dx$

19. $\int e^{\cos x} \sin x \, dx$

20. $\int \cos 2x e^{\sin x \cos x} \, dx$

Appendix D**60 Test Queries for
Performance Evaluation of
Formula Search**

The test queries used for the performance evaluation of the formula search engine is given below. There are two sets of queries that are used to evaluate: (1) training retrieval performance (30 queries); and (2) testing retrieval performance (30 queries). The test queries are organized basically based on the features against which they were tested upon.

30 Test Queries for Training Retrieval Performance Evaluation***Polynomial Functions***

1. $\int (x^3 + x^2 + x + 1) dx$
2. $\int (x^5 + x^4 + x^3 + x^2 + 1) dx$
3. $\int (x+1)(x+2) dx$
4. $\int (x+1)^2 (x+2) dx$
5. $\int (x^2 + 1)^2 dx$

Rational Functions

6. $\int \frac{dx}{x^2}$
7. $\int \frac{dx}{x^2 + x + 1}$
8. $\int \frac{dx}{(x+1)(x+2)}$
9. $\int \frac{x^3 + 1}{(x+1)^2 (x+2)(x+3)} dx$

Appendix D 60 Test Queries for Performance Evaluation of Formula Search

$$10. \int \left((x+1)^3 + \frac{1}{(x+1)^3} \right) dx$$

Irrational Functions

$$11. \int \sqrt{x+1} \, dx$$

$$12. \int x^3 \sqrt{1-x^2} \, dx$$

$$13. \int \frac{\sqrt{x}+1}{\sqrt{x^2+3x+2}} \, dx$$

$$14. \int \frac{1}{\sqrt{(x+1)(x+2)(x+3)}} \, dx$$

$$15. \int \frac{1}{(1+x^2)^{\frac{3}{2}}} \, dx$$

Trigonometric Functions

$$16. \int \sin(2x+a) \, dx$$

$$17. \int \cos x \cos 2x \cos 3x \, dx$$

$$18. \int \sin x \sin 2x \sin 3x \, dx$$

$$19. \int \sin x \cos x \tan x \, dx$$

$$20. \int (x^2+x+1) \sin x \cos x \, dx$$

Logarithmic and Exponential Functions

$$21. \int \ln x \ln(2x) \, dx$$

$$22. \int \ln \ln x \, dx$$

$$23. \int x \ln \ln x^2 \, dx$$

Appendix D 60 Test Queries for Performance Evaluation of Formula Search

24. $\int x^2 \ln \ln x \, dx$

25. $\int \frac{\ln \ln x}{x^2} \, dx$

Mixed Features

26. $\int \ln x \sin x \cos x \, dx$

27. $\int x \ln x \sin x \cos x \, dx$

28. $\int \ln x \sin(\ln x) \, dx$

29. $\int \cos x \ln(\sin x) \, dx$

30. $\int \cos 2x \ln(\sin x \cos x) \, dx$

30 Test Queries for Testing Retrieval Performance Evaluation**Polynomial Functions**

1. $\int (x^3 - x^2 + x - 1) \, dx$

2. $\int (x^6 + x^5 + x^4 + x^3 + x^2 + 1) \, dx$

3. $\int (t+1)(t+2)(t+5) \, dt$

4. $\int (x+5)^2 (x-2) \, dx$

5. $\int (x^3 - 8)^2 \, dx$

Rational Functions

6. $\int \frac{dx}{(x-1)^2}$

7. $\int \frac{5dx}{x^2 - x - 1}$

Appendix D 60 Test Queries for Performance Evaluation of Formula Search

$$8. \int \frac{dx}{(x-1)(x+2)(2x+1)}$$

$$9. \int \frac{1}{(x-1)^2(x+3)^2(x-2)} dx$$

$$10. \int \left(x^3 + \frac{1}{x^3} \right) dx$$

Irrational Functions

$$11. \int \sqrt{2x+5} dx$$

$$12. \int x\sqrt{9-4x^2} dx$$

$$13. \int \frac{\sqrt{x}}{\sqrt{x^2+4x+5}} dx$$

$$14. \int \frac{1}{\sqrt{(x+1)(x+2)(x+3)}} dx$$

$$15. \int \frac{1}{(1+x^2)^{\frac{1}{2}}} dx$$

Trigonometric Functions

$$16. \int \sin\left(4x + \frac{\pi}{2}\right) dx$$

$$17. \int \cos 3x \cos 2x \cos x dx$$

$$18. \int \sin 3u \sin 5u du$$

$$19. \int (\sin x + \cos x) \tan x dx$$

$$20. \int x \sin^{-1} x dx$$

Appendix D 60 Test Queries for Performance Evaluation of Formula Search***Logarithmic and Exponential Functions***

21. $\int \ln(x+1) \ln(2x+1) \, dx$

22. $\int \ln(\ln x + 1) \, dx$

23. $\int 2x \ln \ln 3x^2 \, dx$

24. $\int (x^2 - 1) \ln \ln(x+1) \, dx$

25. $\int \frac{(x+1) \ln \ln x}{x^2} \, dx$

Mixed Features

26. $\int \sin x \ln x \cos x \, dx$

27. $\int (x+1) \ln x \sin x \cos^2 x \, dx$

28. $\int \ln x \cos\left(\ln x + \frac{\pi}{2}\right) \, dx$

29. $\int \sin x \cos(\ln x) \, dx$

30. $\int \cos x \ln(\sin x \cos x \cos 2x) \, dx$

Appendix E

List of Indicative Keywords

The list of 369 indicative keywords that have been used for Text Preprocessing during the preprocessing of mathematical documents is given below.

absolute	circumference	deceleration
accelerate	close	decimal
adjacent	closest	decrease
again	codomain	deduce
always	coefficient	definition
amplitude	collide	degree
angle	collision	depth
approximate	complete	derivative
approximation	concurrent	diagram
arc	condition	difference
area	cone	differential
ascend	connect	differentiate
asymptote	connected	dimension
average	constant	direction
away	contain	displacement
axes	continuous	distance
axis	coordinate	distinct
base	correct	domain
bent	cover	downward
between	critical	draw
bound	cross	drawn
breadth	cube	drop
break	cuboid	each
centre	curve	ellipse
change	cut	enclose
circle	cylinder	equal
circular	cylindrical	equation

*Appendix E**List of Indicative Keywords*

equilateral	gravity	invert
estimate	greater	isosceles
evaluate	greatest	join
exact	ground	large
exactly	heat	larger
expand	height	largest
expansion	hence	leak
express	high	leave
extension	higher	length
external	hospitals	less
extrema	horizontal	lies
extreme	identity	limit
factor	implicit	line
fall	including	linear
family	increase	loci
figure	increment	locus
finish	inequality	lying
finite	infinite	maclaurin
first	inflection	magnitude
fix	inflexion	maximum
flat	initial	measure
flight	inscribe	meet
form	instant	melt
formula	instantaneous	meter
fourth	integer	metre
fraction	integral	midpoint
free	integration	minimum
function	intermediate	minute
furthest	internal	model
general	intersect	moduli
given	intersection	modulus
gradient	interval	moment
graph	inverse	motion

Appendix E

List of Indicative Keywords

move	period	reduce
nature	perpendicular	reduction
near	piece	region
nearest	place	relate
negative	plane	relative
neglect	point	release
negligible	polynomial	remain
never	portion	repeat
normal	position	repetition
number	positive	replacement
obtuse	possible	represent
opposite	pour	rest
order	power	result
ordinate	principle	retardation
origin	prism	reverse
oscillation	project	revolution
overtake	projection	right
parabola	proportional	rise
parallel	prove	rotate
parallelogram	pyramid	rough
parameter	quadrant	rule
parametric	quadrilateral	same
parts	radian	satisfy
partial	radii	second
particle	radius	section
particular	random	sector
pass	range	select
patch	rate	semi
pause	ratio	separate
pendulum	real	series
percent	reason	situation
percentage	rectangle	shade
perimeter	rectangular	shape

Appendix E

List of Indicative Keywords

short	sticky	unchanged
shorter	straight	unequal
shortest	strike	uniform
show	study	unit
side	subsequent	upright
sign	substitution	upward
significant	success	valid
similar	sum	value
simple	surd	variable
simplify	surface	variation
simultaneously	tabulate	vary
sketch	tangent	vector
skew	term	velocity
slope	tetrahedron	verify
slow	theorem	vertex
small	thick	vertical
smaller	third	vertices
smallest	three	volume
solid	time	way
solution	total	when
solve	touch	whole
specified	towards	width
speed	transfer	work
sphere	trapezia	write
spherical	trapezium	written
split	travel	zero
square	triangle	
standard	triangular	
start	trigonometry	
state	turn	
stationary	twice	
steady	two	

Appendix F 60 Test Queries for Mathematical Document Retrieval Evaluation

The test queries used for the performance evaluation of the mathematical document retrieval engine are given below. There are two sets of queries that are used to evaluate: (1) training retrieval performance (30 queries); and (2) testing retrieval performance (30 queries). The test queries are organized based on the typical problem features against with which they were tested upon.

30 Test Queries for Training Retrieval Performance Evaluation

Limits

1. By using the method of a limit, find the gradient function of the following curve
 $y = 3x^2$.
2. Find the limits of the following function when x approaches the given value.
 $\frac{x^2 - 1}{x + 1}$ as $\lim_{x \rightarrow -1}$.
3. Find the limit of the following $\lim_{x \rightarrow -\infty} \frac{x + 1}{x^2 + 3}$.
4. Using the definition, calculate the derivative of the following function
 $r(x) = \sqrt{2x + 1}$.

Differentiation Techniques

5. Differentiate the following with respect to x $2x \tan 3x$.
6. Find $\frac{dy}{dx}$ for the following parametric equations $x = t + 2$, $y = t^2 - 3$.
7. If $y = \frac{2x^2 + 3}{x}$, prove that $x^2 \frac{d^2y}{dx^2} + x \frac{dy}{dx} = y$.

Appendix F 60 Test Queries for Mathematical Document Retrieval Evaluation

8. Use implicit differentiation to find $\frac{dy}{dx}$ of the following $x^2(x-y)^2 = x^2 - y^2$.

Gradients, Tangents and Normals

9. Find the equation of the tangent to the curve $x^2 + 5y^2 + 2xy = 4$ at the point $(1, -1)$.

10. The equation of a curve is $y = (3 - x^2)^6$. Find

(i) $\frac{dy}{dx}$,

(ii) The equation of the normal at the point on the curve where $x = 2$.

11. A curve has parametric equations $x = t^2 - 2t$, $y = t^2 + 2t$. Find

(i) An expression for $\frac{dy}{dx}$ in terms of t ,

(ii) The equation of the tangent to the curve at the point where the gradient is $\frac{1}{2}$,

(iii) The coordinates of the point on the curve at which the tangent to the curve is parallel to the y -axis.

12. Find, to the nearest whole number, the gradient of the curve $y = x \tan x$ at the point where $x = 1$.

Stationary, Maximum and Minimum Values / Points

13. Given that $y = \frac{16}{x^2} + \frac{x^3}{3}$, find the stationary value of y and determine whether it is a maximum or a minimum.

14. Find the x -coordinate, for $0 < x < \frac{\pi}{2}$, of the stationary point on the curve $y = e^{3x} \cos x$.

15. Show that the curve $y = e^x - 2e^{-x}$ has no stationary values.

Appendix F 60 Test Queries for Mathematical Document Retrieval Evaluation

16. Find the x-coordinates of all the stationary points on the curve $y = \frac{x^3}{(x+1)^2}$

stating, with reasons, the nature of each point.

Approximations

17. Given that $y = x^3 + 3x^2$, use calculus to find, in terms of p the approximate percentage increase in y when x increases from 2 by p%, where p is small.

18. A curve has the equation $y = \frac{6}{1-2x}$. Find an expression for $\frac{dy}{dx}$. Hence find

the approximate increase in y as x increases from 2 to $2 + p$, where p is small.

19. Use calculus to determine, in terms of p, the approximate change in the radius of a circle when the area of the circle increases from 900π to $(900 + p)\pi$, where p is small.

Integration Techniques

20. Evaluate $\int \sin 2x \, dx$.

21. Integrate the following with respect to x

$$\int \frac{1}{2x+1} \, dx.$$

22. Express $\frac{1}{(x-2)(x-1)}$ in partial fractions. Hence evaluate

$$\int_3^5 \frac{1}{(x-2)(x-1)} \, dx.$$

23. Show that $\frac{d}{dx} \left(\frac{x}{1+2x} \right) = \frac{1}{(1+2x)^2}$. Hence, or otherwise, evaluate

$$\int_1^4 \left(\frac{2}{1+2x} \right)^2 \, dx.$$

Appendix F 60 Test Queries for Mathematical Document Retrieval Evaluation**Applications of Integration - Area and Volume**

24. Find the exact value of the area enclosed by the parabola $y^2 = 4x$ and the line $y = 2x - 4$.

25. Find, to two decimal places, the volume obtained when the region bounded by the curve $y = 1 + \frac{1}{x}$, the x-axis and the lines $x = 2$ and $x = 3$, is rotated through 360° about x-axis.

26. O is the origin and A is the point on the curve $y = \tan x$ where $x = \frac{1}{4}\pi$. Show that the area of the region enclosed by the chord OA and the arc OA of the curve is $\frac{1}{8}\pi - \ln\sqrt{2}$.

27. A curve passes through the point $P(0, 3.5)$ and is such that $\frac{dy}{dx} = 2 - x$. The normal to the curve at P meets the curve again at Q. Find
 (i) The coordinates of Q,
 (ii) The area of the region between the curve and the line PQ.

Kinematics

28. A particle moves in a straight line so that, at time t seconds after leaving a fixed point, its velocity ms^{-1} is given by $v = \frac{27}{(2t+1)^2} - 3$. Find

- (i) The value of t for which the particle is at instantaneous rest ,
- (ii) The initial acceleration of the particle.

29. Liquid is poured into a bucket at a rate of $60\text{cm}^3/\text{s}$. The volume, $V\text{cm}^3$, of the liquid in the bucket, when the depth of liquid is $x\text{cm}$ is given by

$$V = 0.01x^3 + 2.2x^2 + 200x.$$

Find

- (i) the rate of increase in the depth of liquid when $x = 10$,

Appendix F 60 Test Queries for Mathematical Document Retrieval Evaluation

(ii) the depth of liquid when the rate of increase in the depth is 0.2cm/s .

30. The mass, m grams, of a radioactive substance, presented at time t days after first being observed, is given by the formula $m = 24e^{-0.02t}$. Find the rate at which the mass is decreasing when $t = 50$.

30 Test Queries for Testing Retrieval Performance Evaluation***Limits***

1. Find the limit of the following.

$$\frac{\tan 2x}{x} \text{ as } \lim_{x \rightarrow 0}.$$

2. Using the definition, calculate the derivative of the following function.

$$p(x) = \sqrt{3x}.$$

3. Find the limit of the following.

$$\lim_{x \rightarrow \infty} \frac{7x^3}{x^3 - 3x^2 + 6x}.$$

4. Differentiate the following function from first principles.

$$y = \frac{1}{2}x^2.$$

Differentiation Techniques

5. Differentiate the following with respect to x .

$$\ln(5x+1).$$

6. Find $\frac{dy}{dx}$ of the following.

$$x^2 + y^2 + 2xy - 1 = 0.$$

7. Using the rules of indices, express the following as e^{ax+b} . Hence, or otherwise, differentiate the following with respect to x .

$$e^{3x} \cdot e^{x-1}.$$

Appendix F 60 Test Queries for Mathematical Document Retrieval Evaluation

8. The curve C has the parametric equations $x = f(t)$ and $y = g(t)$. Find $\frac{dy}{dx}$ for the following.

$$x = t^2 - 4, \quad y = \sqrt{t + 4}.$$

Gradients, Tangents and Normals

9. Find the gradient of the curve $y^2 + 2xy = (x - 3)^3 + 16$ at the point (1, 2).

10. Find the value of the constant c for which the line $y = 2x + c$ is a tangent to the curve $4x^2 - 6x + 11$.

11. The equation of a curve is $y = \frac{10}{1 + x^2}$. Find the equation of the normal to the curve at the point where $x = 3$.

Stationary, Maximum and Minimum Values / Points

12.

The diagram shows a piece of wire bent to form the perimeter of the plane shape PQRSTP which consists of a semi-circle and a rectangle. The radius of the semi-circle is x cm and $QR = ST = y$ cm. The length of the wire is 100 cm and the area

of the shape PQRSTP is A cm². Show that $A = 100x - 2x^2 - \frac{\pi x^2}{2}$. Given that x

may vary, show that the maximum value of A is approximately 700.

Appendix F 60 Test Queries for Mathematical Document Retrieval Evaluation

13. A circular cylinder, open at one end, is constructed of thin sheet metal whose area is $432\pi \text{ cm}^2$. The cylinder has a radius of $r \text{ cm}$ and a height of $h \text{ cm}$.

(a) Show that the volume, $V \text{ cm}^3$, contained by the cylinder is given by

$$V = \frac{\pi}{2}(432r - r^3).$$

Given that r can vary,

(b) Find the value of r for which V is stationary,

(c) Evaluate the stationary value of V and determine, with working, whether this value is a maximum or a minimum.

14. Show that the curve $y = e^x - e^{-x}$ has no turning points.

15. Find the coordinates of the stationary points on the curve $y = 27 + 12x + 3x^2 - 2x^3$ and deduce the nature of each of these points.

Approximations

16. Two variables, x and y , are related by the equation $y = x + \frac{5}{x^2}$.

Obtain an expression for $\frac{dy}{dx}$,

and hence find, in terms of p , the approximate change in y as x increases from 4 to $4 + p$, where p is small.

17. A curve has the equation $y = x^3 \ln x$, where $x > 0$.

(a) Find an expression for $\frac{dy}{dx}$,

Hence

(b) find the approximate increase in y as x increases from e to $e + p$, where p is small.

Integration Techniques

18. Evaluate $\int 2 \sin(3x - 1) dx$.

Appendix F 60 Test Queries for Mathematical Document Retrieval Evaluation

19. Express $f(x) = \frac{x^3 + 2}{x^2 - 1}$ in partial fractions.

Hence find the value of $\int_{-4}^{-2} f(x) dx$, Giving your answer correct to 3 significant figures.

20. Express $2 \sin^2 A$ in terms of $\cos 2A$ and hence evaluate $\int_{\frac{1}{2}\pi}^{\frac{3}{2}\pi} 2 \sin^2 \frac{1}{2} x dx$.

21. Given that $\int_1^5 h(x) dx = 4$, evaluate $\int_1^5 [h(x) + 3] dx$.

22. Find $\int \frac{16x + 5}{4x^2 + 2x + 1} dx$ by completing the square and using a substitution to reduce it to standard form..

23. By means of a suitable substitution, or otherwise, evaluate

$$\int_1^3 \frac{x}{(2x-1)^3} dx.$$

Applications of Integration - Area and Volume

24.

The figure shows part of the curve $y = \frac{1}{\sqrt{4x+1}}$.

Find

- (i) the area of the shaded region,
- (ii) the volume generated when the shaded region is rotated through 360° about the x-axis.

Appendix F 60 Test Queries for Mathematical Document Retrieval Evaluation

25.

The diagram shows part of the curve $y = 6 \sin\left(3x + \frac{\pi}{4}\right)$. Find the area of the shaded region bounded by the curve and the coordinate axes.

26.

The shaded region in the diagram is bounded by the curve $y^2 = 2x$, the x-axis and the line $y = 2x - 2$. Find, in terms of π , the volume generated when this region is rotated through 360° about the x-axis.

Kinematics

27. A particle moves in a straight line so that t seconds after leaving a fixed point O, its velocity, $v \text{ ms}^{-1}$, is given by $v = 6 + pt - t^2$, where p is a constant. The particle comes to an instantaneous rest at the point A, 6 seconds after leaving O.

- Evaluate p .
- Calculate the distance OA.
- Calculate the acceleration of the particle at A.

Appendix F 60 Test Queries for Mathematical Document Retrieval Evaluation

28. A particle move is a straight line so that , t seconds after leaving a fixed point O, its displacement , s metres , is given by $s = \frac{1}{3}t^3 - 2t^2 + 3t$. Given that the particle returns to O when $t = T$, find the value of T .

Using this value of T , find

- (i) the maximum displacement from O of the particle during the interval $0 \leq t \leq T$,
- (ii) the acceleration of the particle at time T seconds .

29. A spherical balloon is being inflated in such a way that its volume is increasing at a constant rate of $150 \text{ cm}^3 \text{ s}^{-1}$. At time t seconds, the radius of the balloon is r cm .

- (i) Find $\frac{dr}{dt}$ when $r = 50$,

- (ii) Find the rate of increase of the surface area of the balloon when its radius is 50 cm .

[The formulae for the volume and surface area of a sphere are $V = \frac{4}{3}\pi r^3$,

$$A = 4\pi r^2 .]$$

30. Two variables, x and y , are related by the equation $y = x + \frac{5}{x^2}$.

- (i) Obtain an expression for $\frac{dy}{dx}$,

- (ii) Given that y is increasing at a rate of 2.7 units per minute when $x = 4$, find the rate of change of x at this instant.