

Lexical Error Compensation in Handwritten-based Mathematical Information Retrieval

Seyed Ali Ahmadi

Computer Science Department
George Washington University
alibala@gwu.edu

Abdou Youssef

Computer Science Department
George Washington University
ayoussef@gwu.edu

Abstract. Entering mathematical queries, in general, can be a demanding task. Mathematical notation is two-dimensional and cannot be easily typed with a standard QWERTY keyboard. Handwriting appears to be the most intuitive and promising method to express mathematical queries. Recognition technology for handwritten mathematical notation has never been applied in math search. Existing recognition techniques for handwritten mathematical notation are quite advanced but still produce misrecognized or completely unrecognized symbols. The objective of this research is to design and implement an automated symbol-recognition error compensation system for the handwritten-based query processing. To achieve this objective, we have designed and implemented a query processing algorithm which modifies and expands handwritten queries using different policies, in order to compensate for the symbol-recognition errors. Experiments have demonstrated that these policies substantially improve the performance of a handwritten-based math IR system.

1 Introduction

Typing mathematical queries, in general, can be a demanding task. Mathematical notation is two dimensional and cannot be easily typed with a standard QWERTY keyboard [1]. In order to have an efficient math-aware search system that is capable of locating mathematical expressions and formulas, users should be able to form their queries with the most intuitive and natural method [2]. Providing a math-appropriate query language and user interface that enables users to express their information needs, which often involves math notation, is essential in a math-aware Information Retrieval system [3]. Hand-writing the math notation appears to be the most intuitive and promising method to express mathematical queries. This work introduces a new paradigm for using handwriting as the modality of choice to create math queries and submit them to a mathematical information retrieval system.

Recognition of handwritten mathematical notation has been researched for more than a decade [4-13] but has never been applied to query generation in a mathematical information retrieval system. With handwriting, users are able to write a mathematical formula or expression using a stylus pen and submit it as a query to the math search engine.

No matter what type of recognition technique is used to recognize and process a handwritten query, there will be misrecognized or completely unrecognized symbols as the result of the recognition phase which will decrease the overall performance of a mathematical IR system. Although manual correction of the symbol recognition errors is an option, it will increase the overhead of the search system and decrease user satisfaction. There are also future possibilities for a math search system to receive handwritten queries from another math-aware computer system instead of a human user [14]. Automatic compensation for recognition errors can dramatically reduce the overhead and enable the search engine to communicate with other handwritten-based math-aware systems with the least amount of user interaction.

An automatic error compensation method should be designed to handle the recognition errors of a handwritten query before it is finally submitted to the search engine. The primary objective of this research is to design, implement and test an automated lexical error compensation query processing for a handwritten-based mathematical information retrieval system. Four different error compensation policies are proposed to compensate for lexical errors in

handwritten recognition in which mathematical queries will be modified and expanded before submission to the search engine. The rest of this paper is organized as follows. In section 2 and 3, we review the related work in mathematical information retrieval and mathematical handwritten recognition. The error recovery methods in handwritten recognition and methods of handling recognition errors are covered in section 4. We introduce the lexical error compensation policies for math search in section 6. Section 7 summarizes the experiments and results and section 8 concludes this paper and points to our future directions.

2 Related Work

2.1 Mathematical Information Retrieval

Full-scale and functional Information Retrieval systems, capable of searching for fine-grain mathematical content have started to appear, such as DLMF¹ [2], Mathdex from Design Science [15] and Mathematica's search from WOLFRAM² Research. The primary goal of an IR system in general is to reduce the *overhead* for users in locating the relevant information based on some information need [16]. *Overhead* can be expressed as the time a user spends in all of the steps leading to reading an item containing the needed information (**query generation**, query execution, scanning results to select items to read and reading non-relevant items). Additionally, the immediate research goals of a math IR system are [17]:

1. Enable users to search for both text and mathematical expressions
2. Allow users to express math queries naturally and easily, using common notation

In addition to the conventional search for documents, mathematical search can fulfill higher-level and farther-reaching roles such as [14]: discovery of similarities between fields, computer aided proving, learning aid and selective dissemination of information.

2.2 Digital Library of Mathematical Functions

Search system in DLMF is built on top of a conventional powerful text-based IR system [2, 18]. The following table shows a few queries in DLMF search:

Rendered Form	Query Syntax
$\int_0^{\infty} \sin(\frac{1}{3}t^3 + xt)$	integral_0^infinity sin((1/3)t^3+xt)
$\sqrt{Ai^2 + Bi^2}$	sqrt(Ai^2+Bi^2)
$(\dots)^{(x+2)}$	^(x+2) // (x+2) as an exponent part
$\frac{\dots}{(x+2)}$.../(x+2) // (x+2) as a denominator

Fig. 1. Sample of Text-based Math Queries in DLMF

In DLMF, both math queries and math content are converted to an intermediary language which is the result of *Textualization*, *Serialization* and *Normalization*. Figure 2 is an example of a query which is converted to TextSN.

2.3 Mathdex

Miner et al. [15] have recently proposed an approach for math query formulation and data normalization. This technique which follows the model of mathematical search developed for DLMF, is based on mathematical n-grams and has been implemented in the Mathdex search system. Query generation in Mathdex is through a Graphical User Interface. GUI editors have been used by applications such as Microsoft Equation Editor for mathematical input and have been compared to other input models for mathematics[1].

¹ Digital Library of Mathematical Functions

² <http://www.wolfram.com>

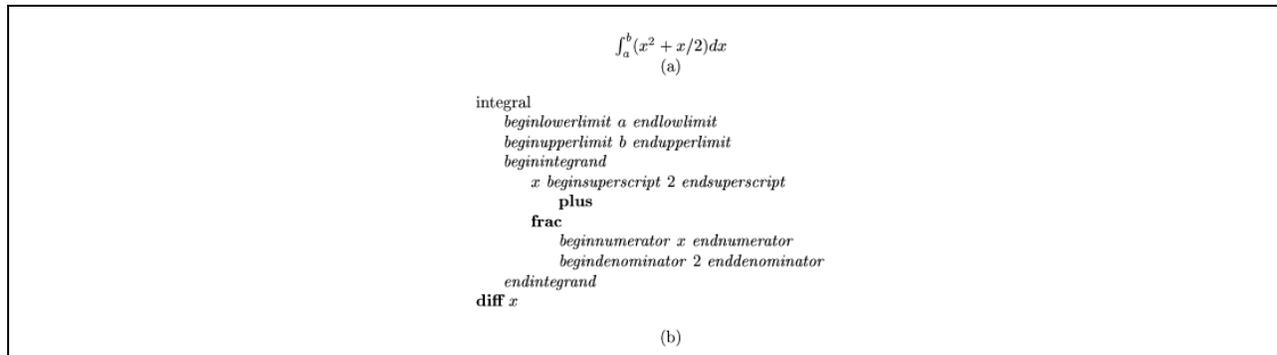


Fig. 2. A query in DLMF, Part (a) Math Expression Part (b) same expression after Textualization + Serialization + Scoped counterpart

2.4 Query Expansion

The proposed solution in this research will use novel query operations to expand a mathematical query in order to compensate for lexical errors in handwritten queries. Traditional text-based search systems use a variety of query operations such as *Query Expansion*, *Term Reweighting*, *Boolean operators* and *Wildcard Replacement* to provide better contextualization and improve their performance [19]. *Query expansion* in text-based systems is usually done through consulting a thesaurus to add semantically similar terms or to use co-occurrence matrices to add terms that usually occur with each other [16]. *Weighted queries* are used to differentiate between terms in a given query. Weighting is the process of assigning an importance factor to terms (either in the query or during the indexing process) to improve the relevance ranking of the search results. In traditional text-based IR systems, *Wildcards* or “*don’t cares*” are used in *Term Masking* as another form of *query expansion*. Wildcards have been used in the text-based queries of DLMF search [20].

2.5 Complexities of Mathematical Notation in Query Interfaces

There has been significant amount of previous research on the complexities of mathematical notation in computer user interfaces [1, 21-23]. This section reviews some of the specific complexities of math notation in building a handwritten-based query front-end for a mathematical information retrieval system:

Large Number of Symbols - Mathematical notation consists of a large number of symbols and characters. In addition to the Latin and Greek alphabet, special characters and operator symbols are used in math notation which will ultimately lead to a more complicated recognition process. In a study by Suzuki [24] for the Infty project, 564 different categories of mathematical symbols were identified based on math expressions in only 25 papers. Unicode 4 supports around 2000 mathematical symbols [25].

Two Dimensional Arrangements - In math notation, symbols and characters are often arranged in a two-dimensional space. The recognition system has to analyze a handwritten query for the two-dimensional structure in a phase called *structural analysis*. The relationship among the symbols in a mathematical expression usually depends on their relative position. On the average, more than 70% of mathematical expressions are two-dimensional [21].

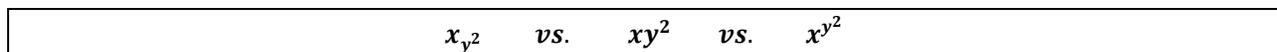


Fig. 3. Importance of relative positioning in math expressions

Grouping of Symbols - Some math symbols are binding symbols (such as parenthesis, braces or square roots) which group other symbols together. There is also *explicit* and *implicit grouping* which introduces a great challenge to structural analysis phase. Also several symbols together may form a special unit (such as trigonometric functions; *sin*, *cos*; Limits; *lim* etc.).

$\int_a^b \frac{\sqrt{a^2 + u^2}}{u^2} du$	$A_0 \left[\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n} \right)^{n \cdot r} \right]$
--	---

Fig. 4. Explicit vs. Implicit grouping of mathematical symbols

Context Dependant Semantics - Math symbols can take multiple meanings based on the context. For example '.' can be a decimal point if it is between digits or it can be a dot product operator; also "dx" can have two different meanings. Some of the proposed recognition methods use the contextual information to improve the recognition.

Abnormal Characters and Symbols - Due to the special nature of the handwritten mathematical notation, symbols and characters might be broken, touching or even both. Uchida et al. categorizes the abnormal characters as: *touching*, *self-touching*, *broken*, *touching and broken* and *overlaid characters*. These abnormal symbols are very challenging in recognition of handwritten query [21].

Non-Uniform Frequency Distribution of Symbols - Frequency of appearance for each category of math symbols varies drastically among documents [23]. In an empirical study by So and Watt, more than 20000 document from the math arXiv³ server were analyzed [26]. The result indicated that in different subjects, different math symbols occur much more frequently than others. This non-uniform frequency distribution can be used in mathematical handwritten recognition as prior knowledge.

2.6 Interaction Modalities for Mathematics

Based on the general objective of an IR system, the most desirable mode of interaction between a user and an IR system is the one which minimizes the *overhead*. Different modes of query input are in fact different ways of building an interface for the query system. In user interface design, after the task analysis is complete and the user actions are identified, the designer can choose from the primary modals of interaction; *menu selection*, *form filling* (text input), *command language*, *natural language*, and *direct manipulation* are among the main categories of interaction modals [27]. Users of math digital libraries must be able to express their search needs in the *native*, concise language of math. The major modals of interaction that have been proposed to communicate mathematical content to a computer system are [1]:

1. text based input forms using a QWERTY keyboard
2. Editor-like GUI programs, such as Microsoft equation editor, using drop-down menus and buttons
3. Handwriting a math equation using a tablet and a stylus
4. Voice-enabled user interface
5. Hybrid modes which are combination of different modals

3 Handwritten Mathematical Recognition

Automatic recognition of handwritten mathematical expressions has been extensively studied and different approaches have been proposed [4-13]. Handwritten mathematical expression recognition usually includes two major phases: *Symbol Recognition* and *Structural Analysis*. Most of the proposed symbol recognition methods are based on the assumption that other preliminary processing steps such as *symbol segmentation* and *layout analysis* have been completed before the recognition phase begins. *Connected Component Analysis* has been extensively used for symbol segmentation [28]. Bounding box extraction, as well as horizontal and vertical projection has been used to isolate the symbols and different components of a document from each other [29]. *Dynamic programming* has also been used for symbol segmentation by optimizing the overall recognition [30].

Due to the potential for pen-based computing, the focus of attention in most of the recent recognition projects has been on *On-line* recognition where mathematical expressions are retrieved from an on-line source such as a tablet or a digitizer in the form of a series of pixel locations along with temporal information (such as [x,y,t] triplets). Some of the previous research focuses on either symbol recognition [8, 31-33] or structural analysis [5, 34-37] while others focus on both [24, 38]. In the second group the final output of the process can be produced in LaTeX or MathML [36].

³ Cornell University Library, <http://arxiv.org>

3.1 Symbol Recognition

Segmentation phase creates objects with some known attributes (such as bounding box location, size and ratio) and an unknown attribute which is the identity of the symbol inside the bounding box to be classified by the symbol recognition phase. Most of the techniques used in symbol recognition are optimization techniques which try to minimize the distance or maximize the correlation between a handwritten symbol and a set of reference symbols.

Elastic Matching [39], *Template Matching* [9], *Structural* and *Neural Networks* [6], *Hidden Markov Models* [40] [41] and *Support Vector Machines* [42] have been used for mathematical symbol recognition. Some of the major proposed techniques are mentioned in a survey [13]. HMMs have been used extensively for speech recognition and can also be used for simultaneous segmentation and recognition of handwritten math symbols.

The set of all possible math symbols is very large [25] and most of the proposed methods only focus on a subset of them [40, 41] or on expressions limited to a certain level of math such as high school math [36]. Different models of combining classifiers, such as *voting mechanisms*, *border count* and *logistic regression*, have also been proposed to better recognize math symbols [43].

3.1.1 Classification Score and Thresholds

A typical symbol recognizer assigns a *classification score* (*cs*) to each candidate for every handwritten symbol in the query. This value can be a probability value, similarity metric or distance metric.

Definition: *Classification Score* is a measure of class membership, assigned by the symbol recognition classifier to every candidate class for any of the handwritten symbols in the query to assess the strength of symbol membership to candidate classes.

Classifiers also use a set of thresholds to differentiate between different levels of membership strength. These levels are categorized as *PASS*, *TIE* and *REJECT*.

3.2 Structural Analysis

The major task of the structural analysis phase is to build a hierarchical structure of the recognized symbols based on the semantics of the math expression. This structure is usually represented by a tree (parse tree or relation tree) called the Symbol Relation Tree (SRT). Most of the proposed methods for structural analysis are based on the assumption that all of the symbols have been completely recognized and their solutions only focus on the structural analysis phase.

4 Error Recovery

Error detection and recovery are essential steps in the design of the query generation system for math IR search. Very few researches have addressed error detection and recovery in handwritten mathematical recognition [44]. Kosmala et al. proposed a manual error correction through a GUI with five new error correction operations: *erase*, *substitute*, *insert*, *undo/redo* and *rewrite*. User will write special symbols for each of these operations on top of the mathematical expression and the recognition system reevaluates the expression [41].

The handwriting quality of mathematical expressions differs greatly among different writers. Chan et al. postulated that it is easier to achieve higher recognition results with neat handwriting and the longer a math expression is, the lower the overall recognition rate becomes. The total number of misrecognitions is often normalized based on the total number of symbols in an expression to reduce the effect of length, but the increasing number of errors in longer expressions still exists even after normalization [44]. Existing mathematical recognition systems have very different performances when all of the different aspects of error handling are taken into consideration. There has been very little research in the literature to recover from the recognition errors. In handwritten mathematical recognition, errors can be categorized as [44]:

- **Lexical errors:** misclassified symbols, due to poor handwriting, similarity between mathematical symbols, or the imperfection of the recognition techniques
- **Syntactic errors:** Errors due to violating the correct syntax of mathematical notation. Example: an arithmetic expression with an unbalanced number of parenthesis

- **Semantics errors:** Errors due to violating the semantics of different math operators. Example: an operator applied to an incompatible operand
- **Logical errors:** Errors in math expression that are syntactically and semantically correct, but logically incorrect; Example: $1+1 = 3$

4.1 Handling Recognition Errors

In a math query system that is based on handwriting, the designer has two options to handle the symbol recognition errors: 1- *Manual Error Recovery* - Asking the user to verify the results of the recognition phase and to manually provide corrections for misrecognized and/or unrecognized symbols, 2- *Automatic Error Compensation* - Automatically handling the errors as much as possible without user interaction. Of course hybrid approaches are also conceivable but the preference is for automatic error compensation.

4.1.1 Automatic Error Compensation: Pros and Cons

In this mode the query system has to handle the errors by itself. Since the query system has no knowledge about the semantics of the expressions at this time, this automatic process is very complicated. This approach has the following pros (+) and cons (-):

- (+) Decreasing the overhead, since the user is not involved in the error recovery
- (+) Possibility of handwritten queries being submitted by another computer system instead of directly by a human user; such as computer algebra systems, automated mathematical proof systems, etc.
A handwritten expression in another math-aware computer system can be submitted to the math search system. For example in proof-aid systems, a math expression can be send to the search engine to verify if such an expression has been used by other mathematicians and so the search result can help continuing the proof. Also in a pen-based computer algebra system, if a handwritten math expression cannot be completely recognized by the system, instead of asking the user to verify it, a math search engine can search for the partially recognized expression and complete it based on the search result.
- (-) Decreasing the precision; since users are not involved, the output of the automatic error recovery system might still have errors and be different from what users really intended as their query; which will lead to a decrease in the precision ONLY comparing to a full manual error recovery. On the other hand, comparing to a math search engine without any support for lexical error handling the automatic system will still improve the performance.

4.1.2 Critical Errors in Handwriting Recognition for Mathematical Search

Not all of the different types of the errors are equally critical in the performance of mathematical IR systems. In fact, what appears to be a syntactic, semantic or logical error, might very well be the exact expression for which the user is searching for. On the other hand, lexical errors are not caused by the user but are due to the inefficacy of the recognition system which will reduce the performance of the search.

4.1.3 Detection of Lexical Errors

Based on classification scores, a classifier passes a symbol, rejects a symbol or considers a tie between two or more of its candidates. If a symbol has been rejected or it is in a tie situation, then there is potential for a lexical error. It is also important to note that a symbol might be considered passed by a classifier but still be a lexical error. For example if a user writes the alpha symbol very much like a lowercase x, there is a good chance that the symbol recognizer classifies it as a lowercase x while the user really meant alpha. Detection of lexical errors therefore is very complicated. This research only focuses on the symbols with classification scores of less than the PASS threshold.

5 Assumptions

Based on the discussions in the previous sections and the review of the related work, the proposed solution for this research satisfies all of the following assumption:

1. Error compensation is applied after the symbol recognition and before the structural recognition.
2. The solution method should cause minimum amount of interaction with users.
3. The solution method only applies to lexical errors; to be precise; it only applies to the symbols with classification scores less than the PASS threshold.
4. The proposed solution should not significantly slowdown the overall performance of the IR system.
5. Symbol recognition and structural analysis are both simulated and not implemented for this research.
6. Lexical errors will be randomly generated in sample queries.

6 Query Operations

Sample queries are written by hand on a digitizer board and the pixel coordinates along with the timing information is generated in an output file. The following diagram is an overview of the proposed math query processing pipeline.

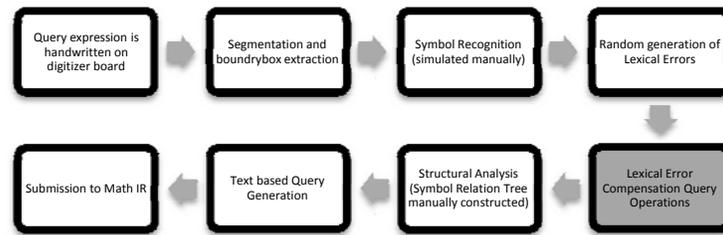


Fig. 5. Query Processing Pipeline

6.1 Symbol Recognition and Structural Analysis Simulation

The technology of handwritten mathematical recognition is quite advanced and there are recognition systems with relatively high recognition rates. Design and implementation of a new recognition system for math symbols is outside the scope of this research. This research simulates the results of such a system, similar to the real results generated by the previously proposed methods. The ground truth UNICODE value of each symbol along with a list of similar symbols, have been manually added to the query files.

In order to have real recognition results from a mathematical handwritten recognition system, we used the *Infty Editor* [24, 45, 46] as one of the successful recognition systems.

6.2 Random Generation of Lexical Errors

Lexical errors are randomly generated in the sample queries. The number of randomly generated errors, expressed as error ratio (ϵ), is a factor of the experiments. Error ratio is defined as the ratio of the number of introduced erroneous symbols to the total number of symbols in a query. The following figure is an example of a math query in which lexical errors have been randomly introduced.

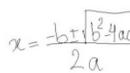
<p>Handwritten Query:</p> 	
<p>correct: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$</p>	<p>with two errors: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$</p>

Fig. 6. Quadratic equation solution, two symbols have been randomly replaced with similar symbols, letter a and alpha, digit 2 and letter z; Number of symbols = 15, $\epsilon = 2/15 \approx 13\%$

Based on the intended error ratio, some of the math symbols will be randomly selected to be replaced by similar symbols to simulate the misrecognition due to similarity between math symbols.

6.3 Lexical Error Compensation Operations

We designed and implemented four different policies to compensate for lexical errors. Before the recognized query is passed on to the structural analysis phase, it will be modified based on one of the following policies:

6.3.1 Policy One – Boolean Expansion Approach

The recognition system proposes a list of possible candidates for symbols whose classification scores are less than the PASS threshold; such symbols are likely lexical errors. Policy one creates an expanded version of the user’s query by applying the logical OR operator and substituting the unknown symbol with all of the proposed candidates by the classifier. This policy will improve the performance by increasing the coverage of search and increasing the chance of retrieving relevant items.

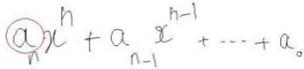
<p>Handwritten Query: </p> <p>Unrecognized Symbol: a Proposed Candidates: 1 - a 2 - alpha</p> <p>Submitted Query: $(a_n x^n + a_{n-1} x^{n-1} + \dots + a_0)$ <OR> $(\alpha_n x^n + \alpha_{n-1} x^{n-1} + \dots + a_0)$</p>

Fig. 7. Policy One - Creating an expanded query based on two symbols proposed for the unknown symbol and applying Boolean OR between math expressions

6.3.2 Policy Two – Iterative submission Approach

Policy two is an iterative approach. The sample query will be submitted to the search engine iteratively and each time one of the proposed candidates for the unknown symbol(s) will be substituted in the query. The motivation behind this approach is that the most probable candidate is more likely to generate the largest hit result list by the search engine and will be selected as the candidate of choice before the final hit results are shown to the user. The order of substitution is based on the *classification score* for each candidate. The candidate with the highest *cs* will be submitted first.

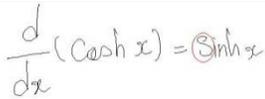
<p>Handwritten Query: </p> <p>Candidates for symbol: first candidate: s (cs_1) second candidate: 5 (cs_2)</p> <p>Assume: $cs_1 > cs_2$</p> <p>Iteration One: Candidate: s Submitted Query: $\frac{d}{dx}(\cosh x) = \sinh x$ number of hits: 110,000</p> <p>Iteration Two: Candidate: 5 Submitted Query: $\frac{d}{dx}(\cosh x) = 5\sinh x$ number of hits: 2</p>

Fig. 8. Policy Two, Math query is submitted to the search engine multiple times and the query with the highest number of hits is selected

6.3.3 Policy Three – Weighted Clause Approach

This policy creates a single query consisting of weighted mathematical clauses; one clause per possible combination of proposed candidates for the unrecognized symbol(s). Policy three uses *classification scores* to calculate a weight assigned to each math clause. The math search system uses these weights for relevance ranking of the hit results. Weighted clause approach helps the search system to rank the most probable clauses higher in the search result.

Relevance ranking is extremely important in information retrieval and has been extensively studied in text-based search systems. Youssef has applied a vector-based model of relevance ranking to DLMF math search as well [47]. The following is an example of a query with one unknown symbol. Weights are calculated based on cs values by a *Weight Calculation Function*.

Handwritten Query:	$\lim_{x \rightarrow 0} \sin \frac{\pi}{x}$
Unrecognized symbol:	π Possible Candidates: 1 - π , cs_1 2 - η , cs_2
Submitted Query:	$(\lim_{x \rightarrow 0} \sin \frac{\pi}{x}) [W1]$ <OR> $(\lim_{x \rightarrow 0} \sin \frac{\eta}{x}) [W2]$

Fig. 9. Policy Three: creating an extended math query based on weighted clauses. cs_1 and cs_2 are the classification scores for π and η respectively. $W1$ and $W2$ are calculated weights of each clause.

6.3.4 Weight Calculation Function

Since the math search engine uses the weights only for the purpose of ranking the hit results before returning them to the user, only the order of the weights are important not their actual values. Hit results that have matched the math clause with higher weigh will be ranked higher in the hit list than those that have matched math clauses with lower weights. In the case of one unknown symbol, weights are directly calculated from the classification scores of proposed candidates. Most of the IR systems assign weights in the [0,1] range, either in the query or in the indexing phase. As an example, if a classifier returns likeliness values from 0 to 100 (such as [48]), the following mapping function f will map them to [0,1] and use them directly as weights:

Handwritten Query:	$\lim_{x \rightarrow 0} \sin \frac{\pi}{x}$
Unrecognized symbol:	π
Possible Candidates:	(1) - π , $cs_1 = 91$ (2) - η , $cs_2 = 63$
Where f is a one-to-one mapping function to [0,1]	
	$W = f(cs) = \frac{cs}{100}$
	$W1 = f(91) = 0.91$, $W2 = f(63) = 0.63$
Submitted Query:	$(\lim_{x \rightarrow 0} \sin \frac{\pi}{x}) [0.91]$ <OR> $(\lim_{x \rightarrow 0} \sin \frac{\eta}{x}) [0.63]$

Fig. 10. Calculation of Weights based on the classification scores, the classifier returns cs values from 0 to 100

6.3.5 Policy Four – Wildcard Replacement

If the classifier is unable to recognize a symbol and the highest classification score is less than or equal to the REJECT threshold, that symbol will be rejected by the classifier. In this case, the classifier has failed to assign the symbol to any of the known classes with enough confidence. The wildcard policy will replace such a symbol with a wildcard symbol. Any math symbol can replace the wildcard symbol during the search.

Handwritten Query:	$A = \int_a^b \frac{1}{2} r^2 d\theta$
	Submitted Query: $A = \int_a^b \frac{1}{2} r^2 d *$

Fig. 11. Policy Four, Rejected symbol has been replaced by a wildcard

6.3.6 Multiple Lexical Errors

All of the examples in the previous sections had only one randomly introduced unrecognized symbol. If the number of the introduced lexical errors in a query is more than one, the policies should be adjusted accordingly. The number

of possible extended queries goes up exponentially as the number of lexical errors in a query grows. If n symbols have been selected to represent lexical errors and the classifier returns r possible candidates for each of them, a total number of r^n different queries can be built. Weights will be calculated based on the sum of classification scores among candidates. The following example shows weight assignment to a math query with two randomly selected symbols as lexical errors.

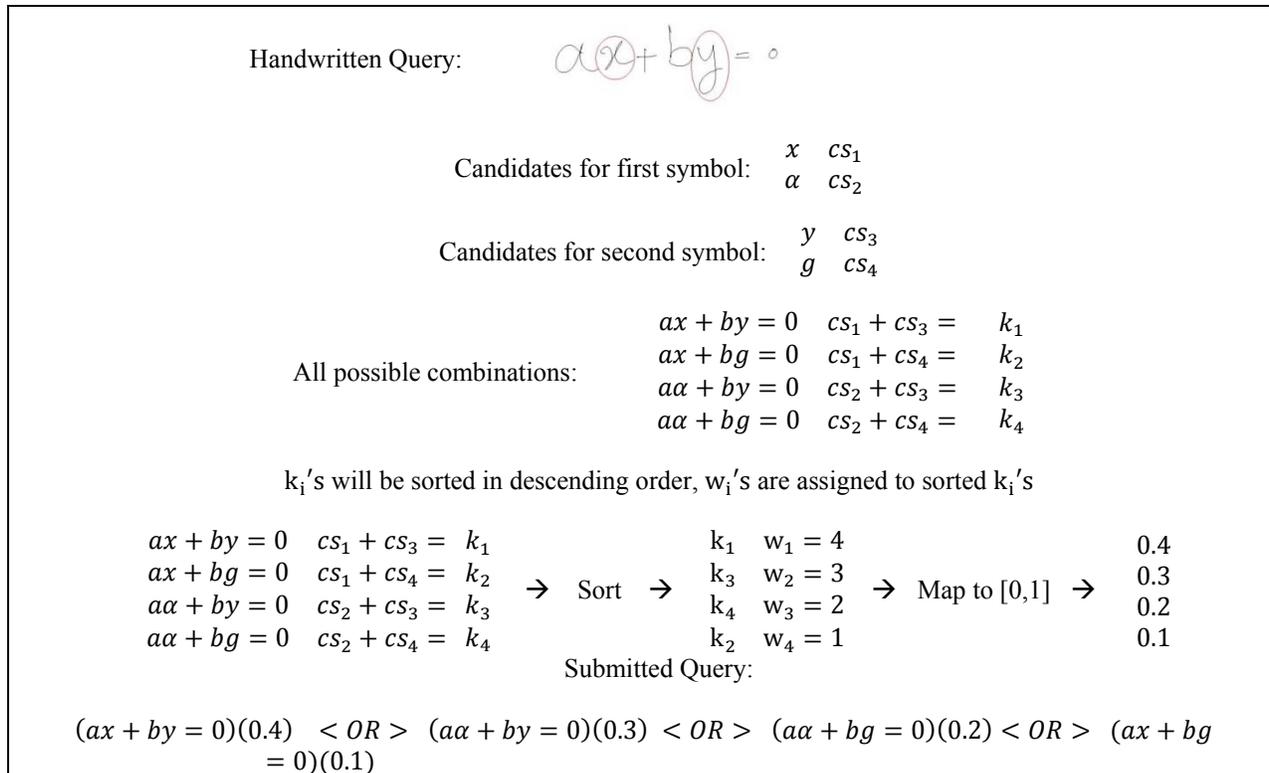


Fig. 12. Mathematical Weighted Clause, Weight calculations with multiple lexical errors

Weights are only used for relevance ranking and their actual values do not change the search results or precision-recall graphs. Higher weights should be assigned to the most probable clause which justifies adding the classification scores of candidates to get the most probable candidate of the classifier for the whole clause.

Although the same method can be applied for more than two lexical errors, in order to keep the scope of the experiments feasible, number of introduced lexical errors and also number of proposed candidates by the classifier have been limited.

6.3.7 Probabilistic Models

Many symbol recognition methods are based on probabilistic models, where the classifier returns a probability for each candidate in the proposed list for a handwritten symbol. If the probabilities of the candidates are independent from each other, then instead of adding the classification scores, the probabilities are multiplied for multiple lexical errors. Figure 13 shows weight calculation in a math query with two randomly chosen symbols and is based on an independent probabilistic classification model.

7 Experiments

DLMF search system was used to conduct the experiments in this research. Since the front-end of DLMF search currently accepts only text-based queries, we designed and implemented a system which processes handwritten-based queries and after applying lexical error compensation policies, converts them to text-based queries, suitable for DMLF math search.

In order to evaluate the effect of handwriting lexical errors in the performance of DLMF search system, a set of sample queries with single math symbols were submitted to the search engine. Each symbol was submitted twice. First the correct symbol was submitted and then as a simulated recognition error it was replaced with a wrong but very similar candidate. The following figure shows some of the symbols that were used in the experiment. Since these symbols are very similar, it is very difficult for a classifier to differentiate them.

Precision and recall were calculated after each submission. Results of the experiment indicate that lexical errors, which are due to symbol similarity, substantially decrease the performance of handwritten-based math search.

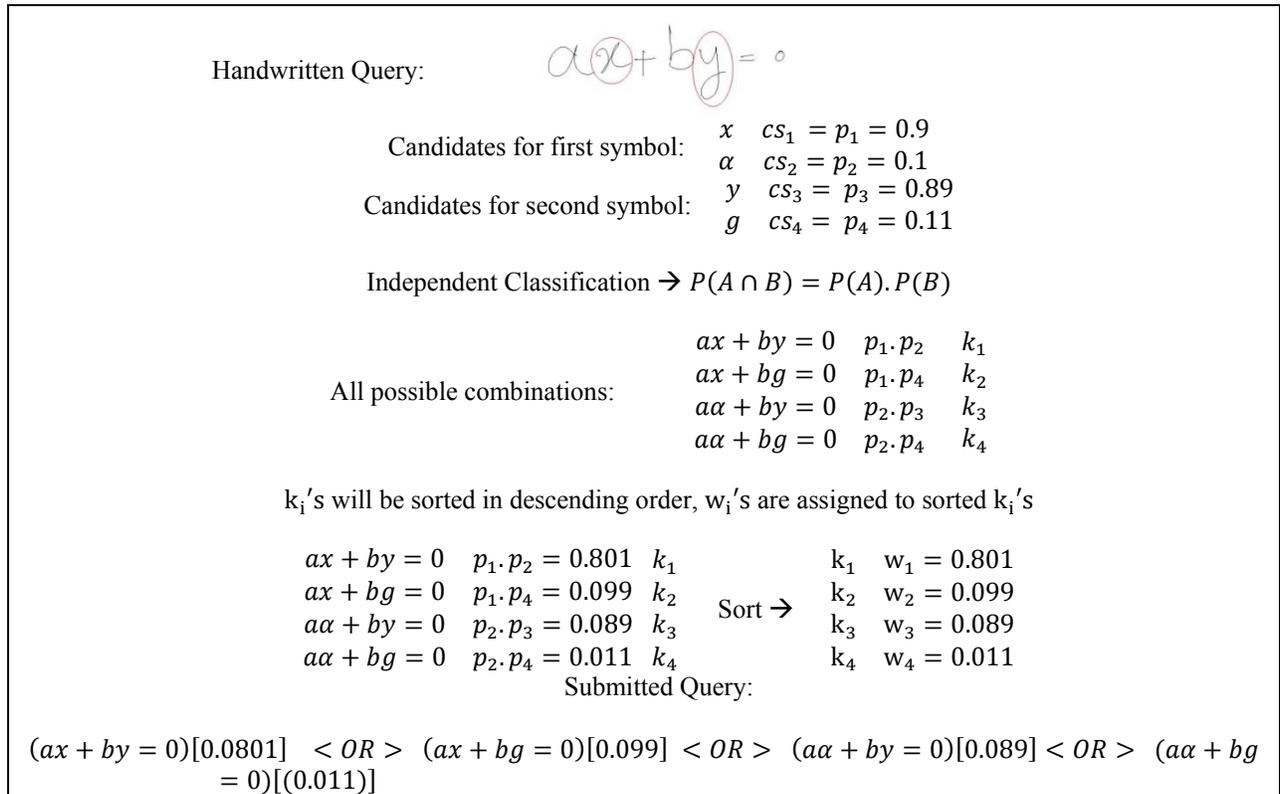


Fig. 13. Mathematical Weighted Clause, Weight calculations with multiple lexical errors for Independent Probabilistic Classification

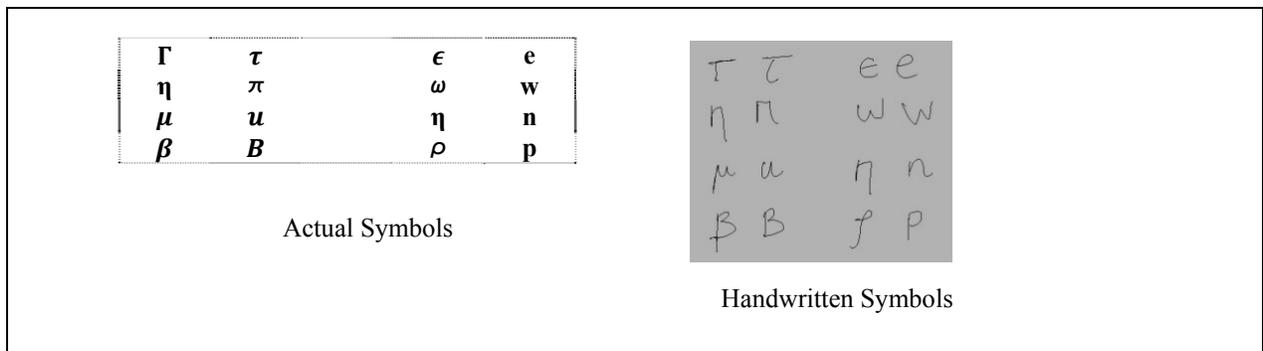


Fig. 14. Example of similar math symbols used in simulated lexical error experiment

We tested the effect of applying policy one to a set of single-symbol sample queries by measuring the precision and recall with and without lexical error compensation.

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q14	Q15	Q16	Q17	Q18
Γ	η	μ	β	ϵ	ω	η	ρ	τ	ψ	α	χ	2	ζ	γ	\sim	α
τ	π	u	B	e	w	n	p	t	w	x	X	z	C	v	∞	∞

Fig. 15. Single Symbol Query Set

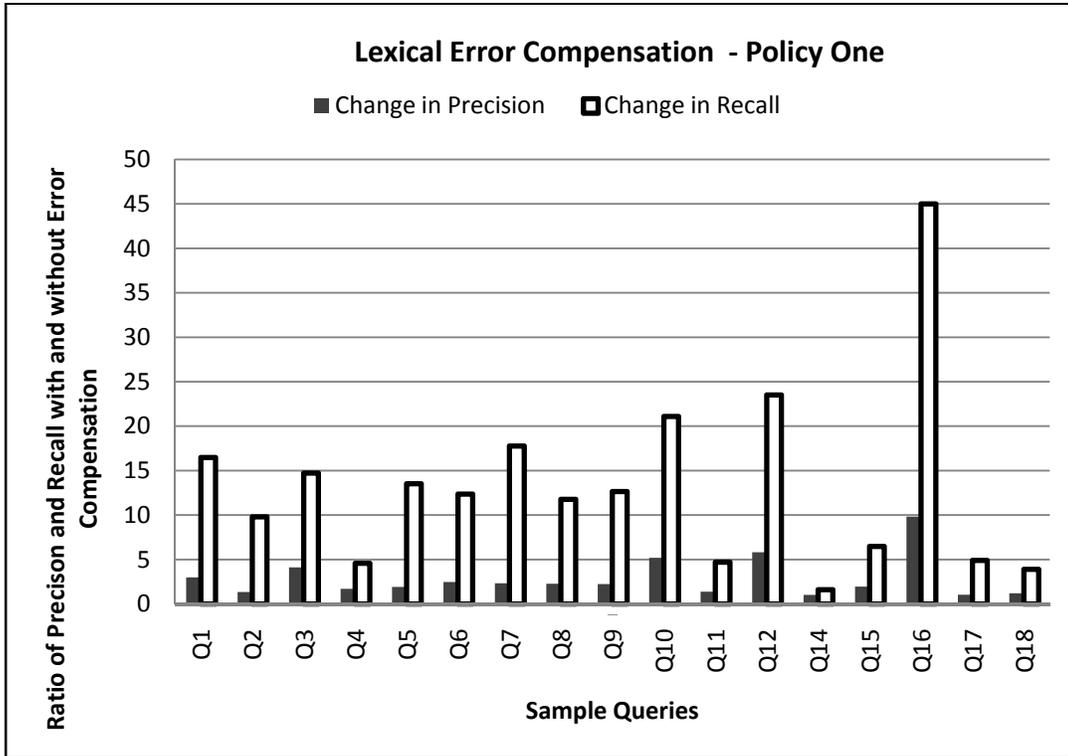


Fig. 16. Lexical Error Compensation – Ratio of Precision and Recall for each query before and after applying Policy one

Applying policy one improved the overall performance of math search by increasing the precision and recall. The average ratio of improvement in precision and recall were 2.87 and 13.22 respectively.

Experiment with Infty

In order to test the effect of the lexical error compensation policies with an existing handwritten recognition system, we used the *Infty Editor*⁴ [24, 45] to create handwritten queries and submit them to the query processing pipeline. Infty Editor is capable of generating LaTeX output for handwritten mathematical expressions. Users are able to use the *Handwriting Math Input Pad* and write their query expressions by hand. Infty will produce a LaTeX string of the recognized output. The following is an example of a handwritten query processed by Infty.

⁴ <http://www.inftyproject.org/>

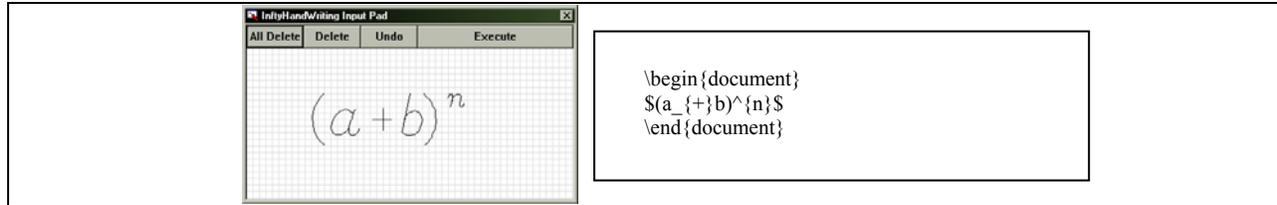


Fig. 17. Sample of a handwritten query recognized by the Infy Editor and converted to LaTeX

A set of small queries with one lexical error were tested in DLMF. Each sample query was submitted twice. Precision and Recall after each submission was measured.

Query Correct Q1	Query with error Q2	# matched items Q1	# matched items Q2	# matched items Q1 <AND> Q2	# matched items Q1<OR> Q2
$(a + b)^n$	$(a \rightarrow b)^n$	18	117	12	18
$a_1 a_2 \dots a_n$	$\alpha_1 a_2 \dots a_3$	3	0	0	3
\sqrt{x}	Vx	70	9	1	78
$\sinh x$	$\sin b x$	172	7	0	179
$\int f(z)$	$s f(z)$	6	2	1	7
\tilde{z}	\bar{z}	2	8	0	10

Fig. 18. Examples of sample queries with one lexical error

We tested the effect of applying policy one to 15 short queries with one lexical error by measuring the precision and recall with and without lexical error compensation. The average ratio of improvement in precision and recall were 3.12 and 11.92 respectively.

8 Discussion and Future Research

In this paper we have presented a new paradigm for using handwriting as the modality of choice to create math queries and submit them to a mathematical information retrieval system. Handwriting enables users of a math search system to create their queries intuitively and with the least amount of overhead. Although handwritten recognition of mathematical expressions is quite advanced and many approaches have been proposed, classifiers still produce misrecognized or unrecognized symbols. We confirmed that these lexical errors substantially reduce the performance of a handwritten-based math search system. We designed and implemented a system for automatic error compensation to handle the recognition errors of a handwritten query before it is submitted to the search engine. Four different error compensation policies were proposed to compensate for lexical errors. We evaluated the effect of lexical errors in math search performance, both by simulation and by using an existing handwritten recognition system (Infy Editor).

As future work we plan to continue the application of handwriting recognition in math search. Changing the front-end of an Information Retrieval system from text-based to handwritten-based is a demanding task and there is substantial amount of research to be done to make sure that it will in fact decrease the overhead for users to find their mathematical needs. Applying the weighted approach requires a close cooperation between an existing handwritten recognition system and the math IR system. We plan to adopt the weighted approach in DLMF search system and experiment with more handwritten queries by using other handwritten recognition systems.

References

1. Zhang, L., Fateman, R.: Survey of User Input Models for Mathematical Recognition: Keyboards, Mice, Tablets, Voice. Tech. Rep., University of California, 2003 (2003)
2. Youssef, A.: Search of Mathematical Contents: Issues and Methods. Gorge Washington University, NIST, Washington, DC (2004)
3. Lorigo, L.: Enhancing Searching of Mathematics. PRL Seminar. Cornell University (2004)
4. Xie, X.: On the Recognition of Handwritten Mathematical Symbols. Computer Science, Vol. Doctor of Philosophy. Western Ontario, London, Ontario (2007) 152
5. Huang, B.Q., Kechadi, T.: A Structural Approach for Online Handwritten Mathematical Expressions. IJCSNS Journal of Computer Science and Network Security **7** (2007)
6. Büyükbayrak, H., Yanikoglu, B., Erçil, A.: Online handwritten mathematical expression recognition. SPIE-IS&T Electronic Imaging **6500** (2007)
7. Watt, S.M., Xie, X.: Recognition for large sets of handwritten mathematical symbols. (2005) 740-744
8. Watt, S., Xie, X.: Prototype Pruning by Feature Extraction for Handwritten Mathematical Symbol Recognition (2005)
9. Suzuki, T., Aoshima, A., Mori, K., Suenaga, Y.: A new system for real-time recognition of handwritten mathematical formulas. Int'l Conf. on Pattern Recognition, Vol. IV (2000) 515-518
10. Chan, K.F., Yeung, D.Y.: Recognizing on-line handwritten alphanumeric characters through flexible structural matching. Pattern Recognition **32** (1999) 1099-1114
11. Miller, E.G., Viola, P.A.: Ambiguity and constraint in mathematical expression recognition. (1998) 784-791
12. Kosmala, A., Rigoll, G.: On-line handwritten formula recognition using statistical methods. Proceedings of the Fourteenth International Conference on Pattern Recognition (1998) 1306-1308
13. Chan, K.F., Yeung, D.Y.: Mathematical Expression Recognition: A Survey. Int'l Journal on Document Analysis and Recognition **3** (2000) 3-15
14. Youssef, A.: Roles of Math Search in Mathematics. Mathematical Knowledge Management (2006)
15. Miner, R., Munavalli, R.: An Approach to Mathematical Search Through Query Formulation and Data Normalization. Lecture Notes in Computer Science **4573** (2007) 342
16. Kowalski, G., Mayburi, T.: Information Storage and Retrieval Systems. Springer - Kluwer Academic (2000)
17. Youssef, A.: Information search and retrieval of Mathematical Contents: Issues and Methods. ISCA International Conference on Intelligent and Adaptive Systems and Software Engineering, Vol. 14, Toronto, Canada (2005)
18. Miller, B., Youssef, A.: Technical Aspects of the Digital Library of Mathematical Functions. Kluwer Academic Publishers (2002)
19. Baeza-Yates, R., Ribeiro-Neto, B.: Modern information retrieval. Addison-Wesley Harlow, England (1999)
20. Altamimi, M., Youssef, A.S.: Wildcards in Math Search, Implementation Issues. 19th International Conference on Computer Application in Industry and Engineering (2007)
21. Uchida, S., Nomura, A., Suzuki, M.: Quantitative analysis of mathematical documents. International Journal on Document Analysis and Recognition **7** (2005) 211-218
22. Fateman, R.: Handwriting + Speech for Computer Entry of Mathematics. UC Berkely (2004)
23. So, C.M., Watt, S.M.: Determining Empirical Characteristics of Mathematical Expression Use. (2005) 15-17
24. Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., Kanahori, T.: INFITY: an integrated OCR system for mathematical documents. Proceedings of the 2003 ACM symposium on Document engineering (2003) 95-104
25. Beeton, B., Freytag, A., Sargent, M.: Unicode Support for Mathematics. Unicode Technical Report. Unicode.Org (2003)
26. e_Print archive. Cornell Univesity (2006) arXiv is an e-print service in the fields of physics, mathematics, non-linear science, computer science, quantitative biology and statistics. The contents of arXiv conform to Cornell University academic standards. arXiv is owned, operated and funded by Cornell University, a private not-for-profit educational institution. arXiv is also partially funded by the National Science Foundation.
27. Shneiderman, B.: Designing the user interface: strategies for effective human-computer interaction. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA (1992)

28. Nomura, M., Uchida, Suzuki: Detection and Segmentation of Touching Characters in Mathematical Expressions. International Conference on Document Analysis and Recognition. IEEE Computer Society (2003)
29. Raja, A., Rayner, M., Sexton, A., Sorge, V.: Towards a Parser for Mathematical Formula Recognition. LECTURE NOTES IN COMPUTER SCIENCE **4108** (2006) 139
30. Toyozumi, K., Yamada, N., Kitasaka, T., Mori, K., Suenaga, Y., Mase, K., Takahasi, T.: A study of symbol segmentation method for handwritten mathematical formula recognition using mathematical structure information. Int'l Conf. Pattern Recognition, Vol. II (2004) 630-633
31. Garain, U., Chaudhuri, B.B., Ghosh, R.P.: A Multiple Classifier System for Recognition of Printed Mathematical Symbols. Int'l Conf. on Pattern Recognition, Vol. I (2004) 380-384
32. Winkler, H.J., Lang, M.: On-line symbol segmentation and recognition in handwritten mathematical expressions. IEEE International conference on Acoustics, Speech and signal processing (1997) 3377-3380
33. Watt, S., Xie, X.: Prototype Pruning by Feature Extraction for Mathematical Handwriting Symbol Recognition. (2005)
34. Tian, X., Fan, H.: Structural Analysis Based on Baseline in Printed Mathematical Expressions Recognition. (2005) 787-790
35. Chan, K.F., Yeung, D.Y.: An efficient syntactic approach to structural analysis of on-line hand written mathematical expressions. Pattern Recognition **33** (2000) 375-384
36. Fukuda, R., Tamari, Ming, Suzuki: A Technique of Mathematical Expression Structural Analysis for the Handwriting Input System. Int'l Conf. on Document Analysis and Recognition, Vol. 5, Bangalor, India (1999) 131-134
37. Chan, K.-F., Yeung, D.-Y.: Towards efficient structural analysis of mathematical expressions. Advances in Pattern Recognition (1998) 437-444
38. Wan, B., Watt, S.: An Interactive Mathematical Handwriting Recognizer for the Pocket PC. MathML conference. University of Western Ontario (2002)
39. Dimitriadis, Y., Coronado, J.: Towards an ART based Mathematical Editor, That uses on-line handwritten symbol recognition. Pattern Recognition **28** (1995) 807-822
40. Kosmala, A., Rigoll, G., Laviolette, S., Pottier, L.: On-Line Handwritten Formula Recognition using Hidden Markov Models and Context Dependant Grammers. Int'l Conf. on Document Analysis and Recognition, Vol. 5, Bangalore, India (1999) 107-110
41. Kosmala, A., Rigoll, G., Brakensiek, A.: On-Line Handwritten Formula Recognition with Integrated Correction Recognition and Execution. Int'l Conf. Pattern Recognition Vol. II (2000) 590-593
42. Tapia, E., Rojas, R.: Recognition of on-line handwritten mathematical formulas in the E-chalk system. Int'l Conf. on Document Analysis and Recognition (2003) 980-984
43. Garain, U., Chaudhuri, B.B.: Recognition of Online Written Mathematical Expressions. IEEE Transactions on Systems, Man and Cybernetics **34** (2004) 2366-2376
44. Chan, K.F., Yeung, D.Y.: Error Detection, Error Correction and Performance Evaluation in On-Line Mathematical Expression Recognition. Pattern Recognition **34** (2001) 1671-1684
45. Fujimoto, M., Kanahori, T., Suzuki, M.: Infty Editor - a mathematics typesetting tool with a handwriting interface and a graphical front-end to openxm. Int'l Conf. on Mathematical Knowledge Management, Vol. 3 (2004)
46. Suzuki M, Tamari F, Fukuda R, Uchida S, Kanahori, T.: Diagram of INFTY. (2003)
47. Youssef, A.: Relevance Ranking and Hit Packaging in Math Search. IMA Workshop. IMA, IMA, University of Minnesota (2006)
48. Eto, Y., Suzuki, M.: Mathematical formula recognition using virtual link network. (2001) 762-767