

Lexical Error Compensation in Handwritten-Based Mathematical Information Retrieval

Seyed Ali Ahmadi and Abdou Youssef

Computer Science Department
George Washington University, Washington, DC, USA
E-mail: alibala@gwu.edu, ayoussef@gwu.edu
URL: <http://cs.seas.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. 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 lexical symbol-recognition errors. Experiments have demonstrated that these policies substantially improve the performance of a handwritten-based math IR system.

Key words: mathematical information retrieval, handwritten mathematical recognition, digital libraries, math search, error compensation

1 Introduction

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]. Handwriting 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. Although there are many proposed methods for recognition of handwritten mathematical notation [3,4], this technology has never been applied to query generation in math search. No matter what type of recognition technique is used, there will be misrecognized or completely unrecognized symbols. Automatic compensation for recognition errors can dramatically reduce the overhead of the search system. 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.

Rendered form:	$\int_0^{\infty} \sin\left(\frac{1}{3}t^3 + xt\right)$	→	Query syntax:	integral_0^infinity_sin((1/3)t^3+xt)
----------------	--	---	---------------	--------------------------------------

Fig. 1. A text-based query in DLMF

1.1 Math Search and Digital Libraries

Full-scale and functional Information Retrieval systems, capable of searching for fine-grain mathematical content have started to appear, such as DLMF [2]. An IR system in general should reduce the *overhead* of the search which 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) [5]. Additionally, a math IR system should allow users to express math queries naturally and easily, using common notation [6]. Search system in DLMF is built on top of a conventional powerful text-based IR system in which both math queries and math content are converted to an intermediary text-based language [2,6]. Figure 1 shows an example of a text-based query in DLMF search.

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]: *text-based*, *GUI editors*, *Handwriting*, *Voice UI* and *Hybrid modes*. We are using handwriting to communicate math queries with the math search systems.

2 Handwritten Mathematical Recognition

Automatic recognition of handwritten mathematical expressions has been extensively studied and different approaches have been proposed [4]. Handwritten mathematical expression recognition usually includes two major phases: *Symbol Recognition* and *Structural Analysis*. Symbol recognition techniques are mainly optimization techniques which try to minimize the distance or maximize the correlation between a handwritten symbol and a set of reference symbols. 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).

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, a similarity metric or a distance metric. Classification Score is a measure of class membership, which is assigned by the symbol recognition classifier to every proposed candidate for any of the handwritten symbols in the query. Classifiers also use a set of thresholds to differentiate between different levels of classification confidence. These levels are usually categorized as *PASS*, *TIE*, and *REJECT*.

2.1 Recognition Errors and Error Compensation

Very few researches have addressed error detection and recovery in handwritten mathematical recognition [8]. 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 [9]. There has been very little research in the literature to recover from the recognition errors. In handwritten mathematical recognition, errors can be categorized as [7]: *Lexical*, *Syntactic*, *Semantic* and *Logical*. **Lexical errors** are misclassified symbols, due to poor handwriting, similarity between mathematical symbols, or the imperfection of the recognition techniques. 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.

A classifier passes a symbol, rejects a symbol or considers a tie between two or more 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.

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. Some of the advantages of the automatic error compensation are:

- 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.

3 Query Operations

Figure 2 shows the query processing pipeline for this research. The proposed query operations are based on the following assumptions:

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.

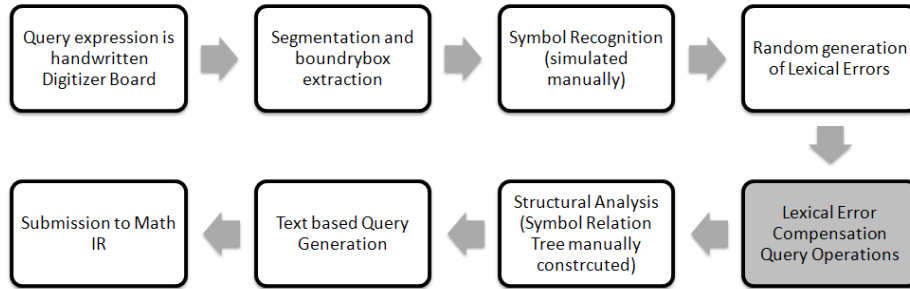


Fig. 2. Query Processing Pipeline

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.

3.1 Symbol Recognition and Structural Analysis Simulation

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* [10] as one of the successful recognition systems. Lexical errors are randomly generated in the sample queries. Selected math symbols will be replaced by similar symbols to simulate the misrecognition due to similarity between math symbols.

3.2 Lexical Error Compensation Policies

1. **Boolean Expansion**—Policy one creates an expanded version of the user's query by applying the logical disjunction and substituting the unknown symbol with the top two proposed candidates by the classifier (Figure 3). This policy will improve the performance by increasing the coverage of search and increasing the chance of retrieving relevant items.
2. **Iterative Approach**—The sample query will be submitted to the search engine iteratively and each time one of the top two proposed candidates for the unknown symbol(s) will be substituted in the query. The most probable candidate is more likely to generate the largest hit result list by the math 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 (Figure 4).

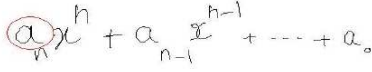
3. **Weighted Math Clauses**—Relevance ranking is extremely important in information retrieval and has been extensively studied in text-based search systems. Youssef has applied a vector-based model of relevance ranking to DLMF math search as well [11]. Policy three uses classification scores to calculate a weight assigned to each math clause (Figure 5). 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. Weights are calculated based on cs values by a Weight Calculation Function.

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 during the indexing phase. Figure 6 shows an example of a classifier that returns likeliness values from 0 to 100, the mapping function f will map them to $[0,1]$ and use them directly as weights. Lexical Errors are due to the symbols that are in the Tie category; whose Classification Scores are less than the PASS and above the REJECT thresholds. In this simulation, the classification score is a random variable in the range of $[PASS, REJECT]$.

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.

3.3 Multiple Lexical Errors

In the case of multiple lexical errors in a query, the compensation policies are adjusted accordingly. The number of possible extended queries can increase 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. However, in order to prevent the exponential growth of query size, the number of possible candidates for each symbol has been limited to at most two. 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

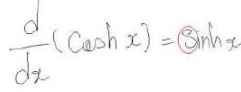


Handwritten Query:

Unrecognized Symbol: **a** Proposed Candidates: 1 - **a** 2 - **α**

$(a_n x^n + a_{n-1} x^{n-1} + \dots + a_0)$ <OR> $(\alpha_n x^n + \alpha_{n-1} x^{n-1} + \dots + a_0)$

Fig. 3. Policy One—Creating an expanded query based on two symbols proposed for the unknown symbol and applying Boolean disjunction between math expressions



Handwritten Query:


Candidates for unknown symbol: 1st candidate: **s** (cs_1) , 2nd candidate: **5** (cs_2)

Assume: $cs_1 > cs_2$

Iteration 1: **s** Submit: $\frac{d}{dx}(\cosh x) = \sinh x$ number of hits: 110,000

Iteration 2: **5** Submit: $\frac{d}{dx}(\cosh x) = 5 \sinh x$ number of hits: 2

Fig. 4. Policy Two—Math query is submitted to the search engine multiple times and the query with the highest number of hits is selected



Handwritten Query:

Candidates for unknown symbol: 1st candidate: **π** (cs_1) , 2nd candidate: **η** (cs_2)

$(\lim_{x \rightarrow 0} \sin \frac{\pi}{x})$ [**W1**] <OR> $(\lim_{x \rightarrow 0} \sin \frac{\eta}{x})$ [**W2**]

Fig. 5. 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, some and some are calculated weights of each clause

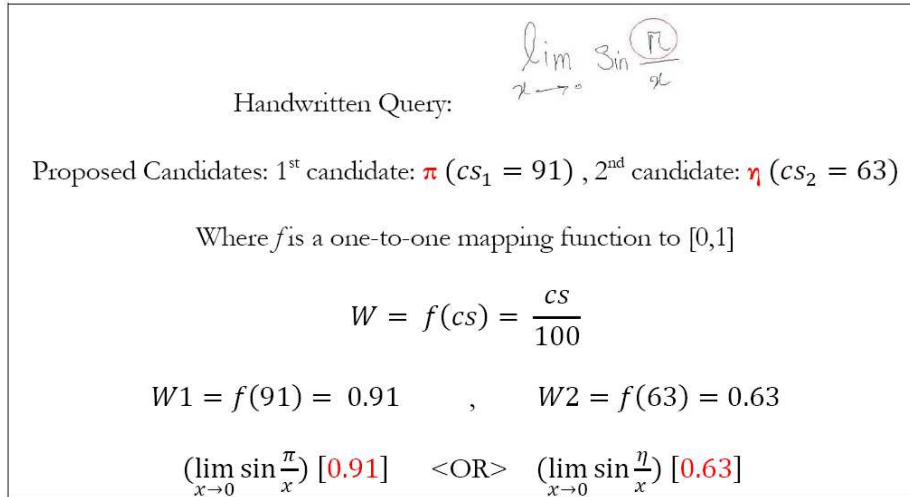


Fig. 6. Calculation of Weights based on the classification scores, the classifier returns cs values from 0 to 100

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. Figure 7 shows weight assignment to a math query with two randomly selected symbols as lexical errors.

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. In independent models, the probabilities are multiplied for multiple lexical errors. Figure 8 shows weight calculation in a math query with two randomly chosen symbols and is based on an independent probabilistic classification model.

4 Experiments

DLMF search system is being used to conduct the experiments in this research. Since DLMF search is currently text-based, we designed and implemented a system which processes handwritten-based queries and after applying lexical error compensation policies, converts them to text-based queries. A set of sample queries with lexical errors were submitted to the search engine in DLMF. Figure 9 and Figure 10 show the set of math symbols that were randomly selected in the sample queries to introduce lexical errors. These randomly selected symbols were replaced with similar symbols to simulate lexical errors. Two separate pilot experiments were conducted on two existing recognition systems to acquire a set of similar proposed symbols for each handwritten

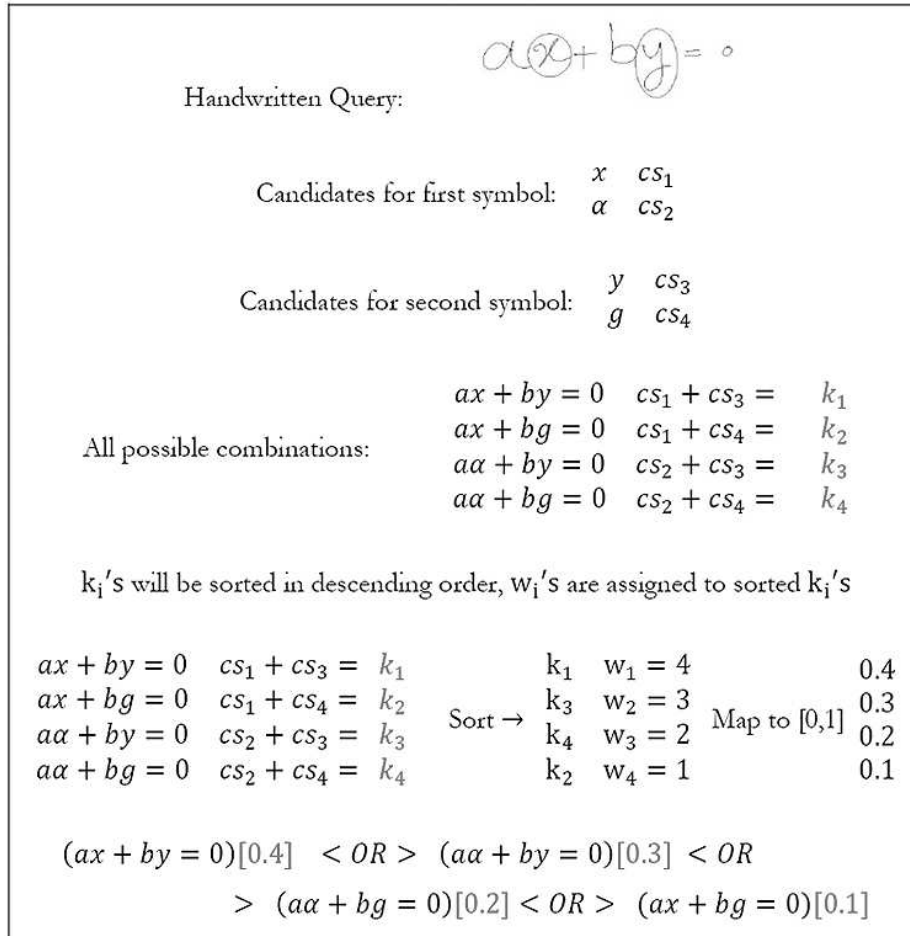


Fig. 7. Mathematical Weighted Clause, Weight calculations with multiple lexical errors

symbol. Queries were then modified with the compensation policies and precision and recall were calculated with and without error compensation. Results of the experiment indicate that lexical error compensation policies substantially improve the performance of handwritten-based math search in DLME. The average ratio of improvement in precision and recall were 2.87 and 13.22 respectively (Figure 11).

In order to test the effect of the lexical error compensation policies with an existing handwritten recognition system, we used the Infty Editor [10] 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

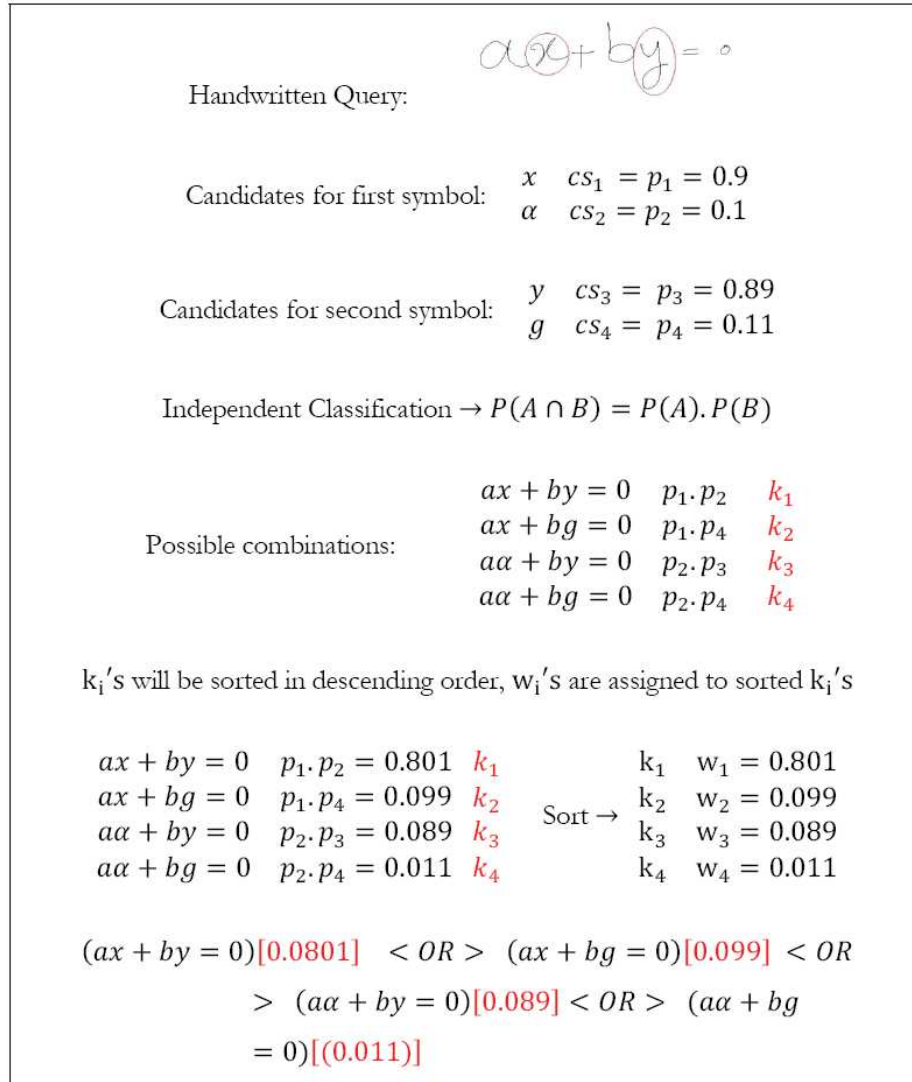


Fig. 8. Weighted Math Clause, Weight calculations with multiple lexical errors for Independent Probabilistic Classification

their query expressions by hand. A set of sample queries were handwritten in Infty and tested in DLMF search engine. Each sample query was submitted twice, with and without error compensation. Precision and Recall with and without error compensation were measured. The average ratio of improvement in precision and recall were 3.12 and 11.92 respectively. Figure 12 shows some of the queries that were written in Infty and were submitted to DLMF search engine.

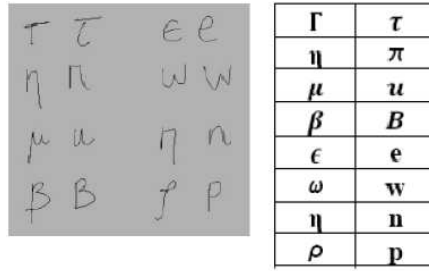


Fig. 9. Similarities between handwritten symbols causing lexical errors

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Γ	η	μ	β	ϵ	ω	η	ρ	τ
τ	π	u	B	e	w	n	p	t
Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	
ψ	α	χ	2	ξ	γ	\sim	α	
w	x	X	z	C	v	∞	∞	

Fig. 10. Similar Math symbols used in sample queries

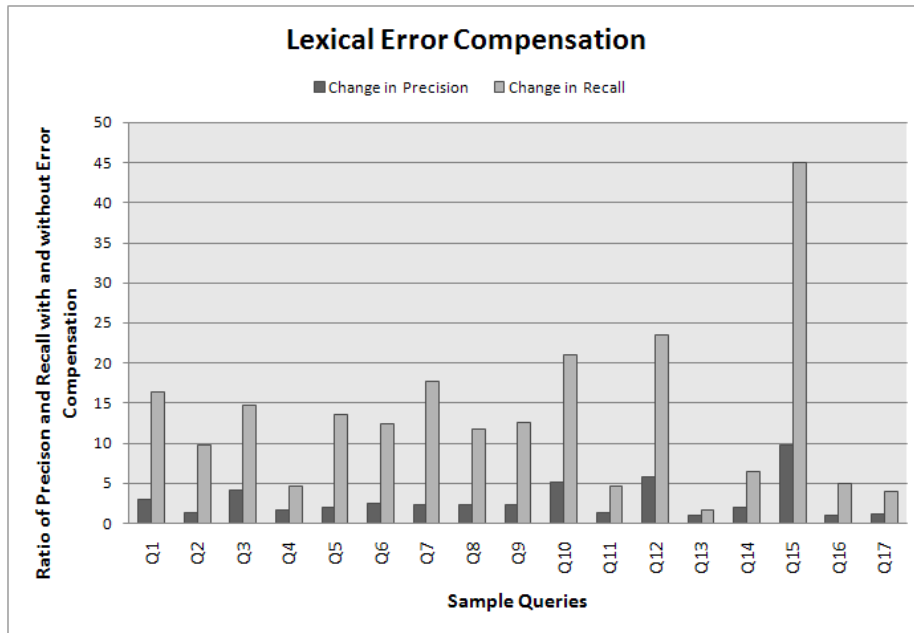


Fig. 11. Lexical Error Compensation — Precision and Recall in DLME, with and without compensation policies

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

Fig. 12. Lexical Errors in sample queries, generated by Infty and tested in DLMF search with and without error compensation

5 Discussion and Future Research

In this research 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. 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 lexical errors of a handwritten query before it is submitted to the search engine. We evaluated the effect of lexical errors in math search performance, both by simulation and by using an existing handwritten recognition system (Infty Editor). As future work we plan to continue the application of handwriting recognition in math search. Changing the front-end of a mathematical 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: Keyboard TabVoice. Tech. Rep., University of California (2003).
2. Youssef, A.: Search of Mathematical Contents: Issues and Methods. GWU, NIST, Washington, DC (2004).
3. Watt, S. M., Xie, X.: Recognition for large sets of handwritten mathematical symbols. (2005) 740–744.
4. Chan, K. F., Yeung, D. Y.: Mathematical Expression Recognition: A Survey. IJDAR 3 (2000) 3–15.

5. Kowalski, G., Mayburi, T.: Information Storage and Retrieval Systems. Kluwer (2000).
6. Youssef, A.: Information search and retrieval of Mathematical Contents: Issues and Methods. ISCA, Vol.14, Toronto, Canada (2005).
7. Miller, B., Youssef, A.: Technical Aspects of the Digital Library of Mathematical Functions. Kluwer Academic Publishers (2002).
8. 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.
9. 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.
10. Fujimoto, M., Kanahori, T., Suzuki, M.: Infty Editor — a mathematics typesetting tool with a handwriting interface and a graphical front-end to epenxm. Int'l Conf. on Mathematical Knowledge Management, Vol. 3 (2004).
11. Youssef, A.: Relevance Ranking and Hit Packaging in Math Search. IMA Workshop. IMA, IMA, University of Minnesota (2006).