Workshop Proceedings of the 8th International Conference on Intelligent Environments
J.A. Botía et al. (Eds.)
IOS Press, 2012

doi:10.3233/978-1-61499-080-2-286

Intelligent Assistive Interfaces for Editing Mathematics

Dilaksha ATTANAYAKE[a], James DENHOLM-PRICE[b] , Gordon HUNTER[b,1], Eckhard PFLUEGEL[a], Angela WIGMORE[a,c]
[a] *School of Computing & Information Systems*
[b] *School of Mathematics*
Kingston University, KT1 2EE, U.K.
[c] *ESS Ltd., U.K.*

Abstract. In this paper, we discuss the need for improved user interfaces for editing mathematical text, focusing of three types (individuals suffering from various disabilities, people relying heaving on on-line learning systems and ones relying on using portable devices) of people for whom conventional editing approaches are not very satisfactory. After reviewing various recent approaches, we focus on describing the development and evaluation of our own intelligent web-based interfaces, *TalkMaths* and *SWIMS* for editing mathematical text. The former is a speech-based editing interface, the latter a system which assists the user through the predictive and corrective power of statistical language models. It offers options for predicting what will appear next (analogous to predictive text for SMS messages) and identifying likely errors due to simple mistakes on the user's part in order to assist in correcting the errors. Using text-stream input, we investigate the utility of the error identification by studying the proportion of times the correct version of the complete mathematical expression appears within the M most likely alternatives suggested by our system. These systems are currently independent of each other, but we aim to integrate the facilities they provide into a simple intelligent assistive interface.

Keywords. statistical language model; web-based mathematical editors; assistive technology

Introduction

Information and Communication Technology (ICT) has been having a greater and greater impact on education, in the classroom and elsewhere over the last few decades. For example, nowadays, teachers can use smartboards and similar facilities to annotate notes in front of their class, then save the resulting "hybrid" document for their students to download and review in their own time. This greatly reduces the burden on students for taking accurate notes at high speed during classes, which was normal 20 years ago.

However, in some ways, mathematics – which is a core subject of study in the school curriculum in most countries and proficiency in which, at least at an elementary level, is essential for success in a wide range of scientific, technical and commercial fields - is perhaps much less suited to be taught via modern educational ICT than many other disciplines. It is a subject which many students find difficult, partly due to its specialized language and notation. These make working with mathematical equations and formulae a problem for a large proportion of people. This is even more notable when the mathematical expressions to be manipulated are to be included in electronic documents. Typing and editing ordinary text can be both slow and error-prone for non-experts, and this is even more the case for mathematical text, with its non-alphanumeric symbols and typically somewhat complicated two-dimensional layout. Furthermore,

[1] Corresponding Author: G.Hunter@kingston.ac.uk

creating, editing and reading mathematical text (in its conventional form) is particularly difficult for three types of groups of people : individuals suffering from various disabilities [5], people relying heavily on on-line learning systems [24, 25] (particularly distance-learning students), and people relying on using portable devices, such as smartphones and tablet computers, to access learning resources. These factors can severely limit the educational and career opportunities of such groups.

In this paper, we first review some existing recent approaches to addressing these problems. We then focus on our own approaches, namely using spoken input as an option for creating and editing mathematical text in electronic documents, and providing an "intelligent assistant" to aid the user by predicting what may come next and offer a semi-automatic correction facility to rectify mistakes.

Spell checkers, automated correcting facilities and predictive text have been familiar features of word processing and text messaging systems for a number of years. These have aimed to provide "intelligent assistance" to the user in order to make the task of creating and editing ordinary text easier. In this paper, we discuss the development of similar features for an editing system for mathematical text. Although the prototype prediction and correction system described here is a "proof of concept", it is proposed to integrate it with our existing editor, TalkMaths [6], which now supports both spoken and typed input and editing commands, and is available online [6].

1. Problems Accessing and Creating Mathematical Content

As noted above, mathematical text tends to have a complicated layout, with a notation which is not simply related to "regular" natural language. Complicated and/or multi-dimensional mathematical expressions cannot easily be conveyed in narrative terms [26]. Österholm [27] performed a study to find out if reading a mathematics textbook with symbols required different skills from reading one without symbols. He concluded that, whilst the use of non-alphanumeric symbols gave mathematics great strengths, saving time and space, comprehending texts using this specialised language and notation, and also translating written words into algebraic equations, required particular skills which students needed time and considerable effort to acquire [27]. Furthermore, established ways of creating, formatting and editing mathematical text in electronic documents – including LaTeX and MathML, and even the GUI-based equation editors embedded in many modern word processors – are not particularly easy for novices to learn to use. These issues are even more relevant for the three types of people mentioned previously : individuals with many types of disability [5], on-line and distance learners [24, 25], and people relying on portable devices with small screens and keyboards which make accessing and typing mathematical content very difficult indeed. Some previous authors have tried to address these issues by taking novel approaches. We briefly review some of these here.

2. Previous Approaches to Addressing the Problems

2.1. Solutions using Tactile Output

Braille has been a successful tactile medium for nearly 200 years, enabling the blind to read and write. Conventional Braille codes use a two-dimensional representation for each alphanumeric character, but these are arranged in a "linear" format, much like the normal orthography of English, to represent words, phrases and sentences. This is not particularly well-suited to represent mathematics. However, novel extensions and modifications to Braille have allowed blind and other visually-impaired students a

much wider access to mathematical resources (e.g. [37]). Nevertheless, all of these coding schemes need to be learned by both the students and teachers, which is a problem for most teachers since they (except specialist teachers of the blind) will probably only teach a rather small number of blind students over many years.

2.2. Solutions using Optical Character Recognition (OCR)

For some groups, typing, but not writing, mathematics is a problem. These would include people with certain types of repetitive strain injuries, people using small mobile devices and, in some cases, on-line distance learners [25]. For such people, an appropriate option might be to write the necessary mathematical expressions using a smart pen or stylus. The characters used could then be identified using an optical character recognition system, and converted into (correctly) typeset mathematical text in electronic form. Some previous authors [38, 39] have developed systems following this approach. However, two remaining issues are how to deal with the possibilities of misidentified symbols (potentially a big problem, since many people have poor "on-screen" handwriting) and mistakes by the user. Previous researchers have used syntactic [40] or statistical [41] approaches in attempts to resolve these issues. The latter approach is to some extent similar to the methodology we use in this paper (see section 4.1 below).

2.3. Solutions using Head Motion or Eye Gaze Direction Monitoring

Although we are not aware of these approaches having been applied to the editing of mathematical text, severely disabled people, including tetraplegics, can interact with computers using systems which monitor motions of their head, eyes, or possibly facial muscles [44]. One such system of particular note is *Dasher*[43], which uses statistical language models (see section 4.1 below) to allocate an appropriate area of a display screen according to the likelihood of the character, word or symbol displayed there being the next item in the sequence being input. *Dasher* can be controlled using a mouse, pointer or any motion or graze tracking system. It should be possible to adapt *Dasher* for use with mathematical editors.

2.4. Solutions using Spoken Input and/or Output

Several groups, including the visually impaired, people with limited use of their hands or arms, and people using portable devices, could benefit by the input and/or output modalities being through speech. There have been a variety of systems attempting to provide synthetic speech descriptions of mathematical text, including *AsTeR* (*Audio Systems for Technical Readings* [31], *MathGenie* [42], *REMathEx* [28], the commercial system *MathPlayer*™ [29], and *AudioMath* [36]. The latter system is open-source, but unfortunately only functions in Portuguese. Previous approaches to allowing spoken input of mathematics include the research prototype systems of Bernareggi & Brigatti [30] (which only works in Italian) and Hanakovič and Nagy [34] (which is restricted to use with the Opera web browser), plus the commercial systems *MathTalk*™ [32] (which is only compatible with certain commercial editiors) and *Math Speak & Write* [33] (which has a rather limited mathematical vocabulary). All of these systems allowing spoken input of mathematics have serious limitations, prompting us to develop our own system, *TalkMaths*.

3. The TalkMaths System

3.1. Overview

The *TalkMaths* system initially started as a desktop application for creating and editing mathematical text, but allowing input by speech alone [4]. With recent developments to the project, the current *TalkMaths* system is now a web-based application [3] and the additional facility for typing input has been added to make the system more useful to a wider audience. The system currently uses a commercial speech recognition system as a front-end. However, since *TalkMaths* works by using a context free grammar to parse a text stream resulting from the speech recognition process, other speech recognition systems could also be employed. Several different editing paradigms (see Figure 1) proposed as a result of earlier work [4, 15] have also been incorporated into this new web-based solution. Three such methods, highlighting all sub-expressions, all individual symbols and all operators, respectively, are illustrated in Figure 1. The appropriate box, corresponding to a particular sub-expression, of the user's choice can then be selected for further editing by specifying the number indexing it.

Figure 1. Different editing paradigms for editing mathematics by speech.

3.2. Speaking Mathematics

Attempts at providing standards for speaking mathematics have been given by various previous authors [16, 14]. These have been specified as formal languages aiming to model, to as great an extent as possible, the natural spoken language constructs that people may use when dictating or teaching mathematics. Our approach, slightly extending that of [14, 4], has been to design our formal language in order to be as close as possible to how mathematically proficient people speak or read mathematical equations and formulae. This should make the formal language relatively easy to learn and use but certain compromises have to be made to avoid potential ambiguities. In particular, for dictating single alphabetical characters (a-z), the names from the NATO pronunciation alphabet [12, 15] must be used. An example of a simple mathematical expression is the equation for velocity under uniform acceleration $v = u + at$ which in our spoken mathematical language would be read as: "*victor equals uniform plus alpha tango*". A more complex example is the formula for the solutions of a general quadratic equation:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

which would be spoken as "*minus bravo plus or minus square root of bravo squared minus four alpha charlie all over begin two alpha end*". Greek characters, such as α, β, etc. can be inserted using the prefix "greek" before the name of the character. For example, the trigonometric identity : $sin(\alpha + \beta) = sin\ \alpha\ cos\ \beta + cos\ \alpha\ sin\ \beta$ would be read as "*sine begin greek alpha plus greek beta end equals sine greek alpha cos greek beta plus cos greek alpha sine greek beta*".

3.3. User Evaluation of TalkMaths Version 1.0

The original version of *TalkMaths* was tested on a group of users, none of whom had any disability. The results of that evaluation, which indicated that non-disabled users, who were more used to using the keyboard and mouse than a speech recognition system, were faster and made fewer errors when creating and editing mathematical text

using a conventional editor than when using *TalkMaths* [4, 15]. However, they did find learning to use *TalkMaths* interesting and relatively straightforward. We also had one participant who suffered from Duchenne Muscular Dystrophy and was wheelchair-bound. He was able to use the keyboard and mouse, but found this unpleasant, and he did have some previous experience of using automatic speech recognition systems. He performed much better using *TalkMaths* than did the non-disabled group, and on many tasks was faster using our system than he was with the conventional editor [4]. This illustrates the potential of *TalkMaths* as a useful tool, particularly for people for whom conventional types of interface are not very satisfactory.

4. New Intelligent Features for Prediction and Correction

4.1. Background

A wide variety of existing technological systems employ prediction and/or correction methodologies in an attempt to make the systems more useful and usable. These include automatic (or semi-automatic) correction systems found in word processors and internet search engines ("Showing results for … Search instead for …") and the prediction systems used in Automatic Speech Recognition (ASR) systems and SMS text message editors on mobile telephones. Although manufacturers of commercial products rarely reveal exactly their secrets, it is understood that correction systems look for "close matches" to what was entered from a database of common words or phrases, whilst prediction systems use statistical models. These models give probabilities of words and word sequences, using information from a large set of previously observed data and evidence from the current situation together with an "inference rule", such as a Bayesian framework, in order to combine information from more than one source [9].
It has been noted that the majority of human typing and spelling errors are quite minor, often involving just the omission or addition of a single character, typing two characters in the wrong order, or accidentally substituting one character for another (often one adjacent to the correct symbol on the keyboard). The Damerau–Levenshtein distance [1, 2] between two character strings measures how different the strings are by taking account of the minimum numbers of insertions, deletions, substitutions and transpositions of characters required to transform one of the strings into the other. Although one of the original motivations for the development of this metric was to compare the similarity of short pieces of natural language text, it has also been applied in fields such as genetics, for example to study how similar two fragments of DNA are to each other. However, we believe that our present paper is the first application of this metric to descriptions of mathematical expressions.

Statistical language models (SLMs) [18] have been at the core of ASR systems for many years [9, 10] where they use statistics from "past experience" to predict the likelihood of what will be spoken next, and combine this with evidence from the acoustic signal of the speech to decide what words were actually said. More recently, such SLMs have been incorporated into innovative systems for automatic translation between languages, such as Google Translate [11].

The simplest types of SLMs are N-gram models, which use statistics of the occurrences of specific sequences of N consecutive words within a database (or "corpus") of training material observed in the past. Individual words (N=1) are referred-to as unigrams, pairs of consecutive words (N=2) as bigrams and triplets of consecutive words (N=3) as trigrams (see [19] for more details on use of trigram and similar models). Longer N-grams are not commonly used [9, 21]. N-gram models can be used both for analyzing how likely or common an observed sequence of words is

(for example, is a given piece of text more typical of author A or of author B?), or for predicting the most likely candidate words or words to next appear in a sequence. This latter case is used in both ASR systems and in predictive text systems.

4.2. Datasets and Building the SLMs

Some ASR systems require the user to enroll (the process of adapting the system to a specific user, before it can actually be used for speech). In order to do this, a user has to provide samples of his or her speech input, which is then used to train the ASR. However, these samples do not cover all possible speech patterns and hence speech recognition programs tend to find "out of vocabulary" words or improbable word sequences when put to work. We hope our predictive and corrective language models will impose constraints which can be efficiently used in combination with those language and acoustic models built-in to the ASR system to correct these errors, with the aid of minimal intervention by the user, as perhaps the best source of knowledge on what was actually said by the user is the user him/herself.

For the work presented in this paper, we built various trigram-based SLMs for spoken mathematics generated from content extracted from a variety of "tutorial" web sites on elementary and intermediate level mathematics. We obtained approximately 4100 equations from such web sites and converted these to the simplest equivalent "spoken" forms using the formalized language described in Section II B. Table 1 shows the most frequent words found in this corpus of spoken mathematics. The SLMs were built from this data using the CMU Toolkit [7], applying Good-Turing discounting [13] and then evaluated for perplexity, as in our previous work [8]. The complete set of distinct words found in the corpus formed the vocabulary of the system.

Table 1. The most frequent words in our "spoken mathematical expressions" corpus. "x-ray" is the spoken form of the symbol "*x*".

Word	end	begin	x-ray	of	two	power	equals	bracket
Frequency %	7.58	7.57	7.3	7.26	4.82	3.93	3.79	3.61

5. *SWIMS* Prototype System

In this section, we introduce our prototype web-based mathematical document editor, *SWIMS (Speech-based Web Interface for Mathematics using SLMs)*, as a proof of concept. *SWIMS* was developed as a separate module which can be later integrated into the *TalkMaths* system following successful evaluation. The goal of *SWIMS* is to assist the user by predicting and/or correcting his/her input using SLMs prior to parsing, required in order to display the output on the screen using suitable mathematical rendering technology such as *MathML*. For ease of evaluation and for better performance, *SWIMS* has been divided into two units, one to predict the next word(s) in the input and the other to correct user mistakes. The former interface is called "Predictive Mathematics" and the latter "Alternative/Corrective Mathematics". We used *JSON* to store trigram probabilities for our SLM, *JavaScript* for calculating probabilities, and *jQuery* library to communicate between the browser and the *TalkMaths* parsing server. See [8] for a high-level diagram of the system's architecture.

*5.1. Predictive Mathematics Inter*face

The predictive mathematics interface predicts one or two words ahead of the currently typed or dictated mathematical text. In order to predict one word ahead, the system uses the last two words of the input to match trigram probabilities. In the case that the input is less than two words or there is no matching trigram, then the system will back-off

[17, 20] to bigram probabilities and if this is not successful, unigram probabilities will be used. Two word prediction is a recursive extension of the one word prediction mechanism. Figure 2 shows this applied to the formula for the voltage across a capacitor which is being discharged through a resistor.

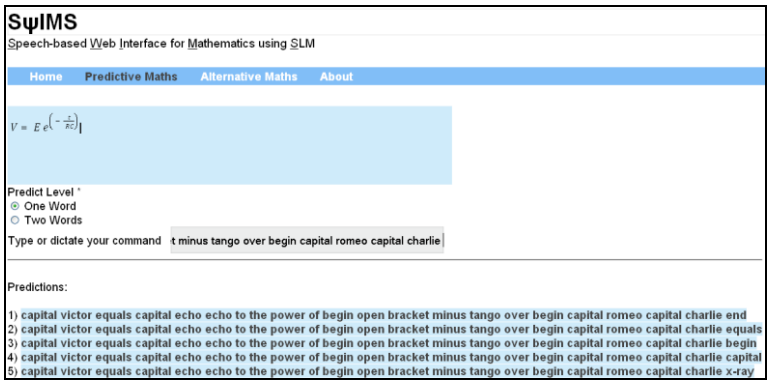

Figure 2. Predictive Mathematics Interface in use. In the top-ranked suggestion, the SLM predicts that "charlie" will be followed by "end".

5.2. Alternative/Corrective Mathematics Interface

To realize correction of errors, we implemented another web interface called Alternative/Corrective Mathematics. Once an "out of vocabulary" (OOV) word is detected, the Damerau – Levenshtein algorithm [1, 2] is used to calculate the Levenshtein distance of the typed word relative to each word in the vocabulary, in order to find suitable candidates for correction of the OOV word in question. Once a list of such candidates has been obtained, SLM probabilities can be used to re-rank the resulting new sequences of words. To illustrate this concept, we designed three variants of correction methods in the Alternative/Corrective Interface of SWIMS. These use Damerau – Levenshtein only, SLM only, and both in combination, respectively. To date, only the first of these has been developed and tested, but we intend to implement and evaluate the other two methods in future work.

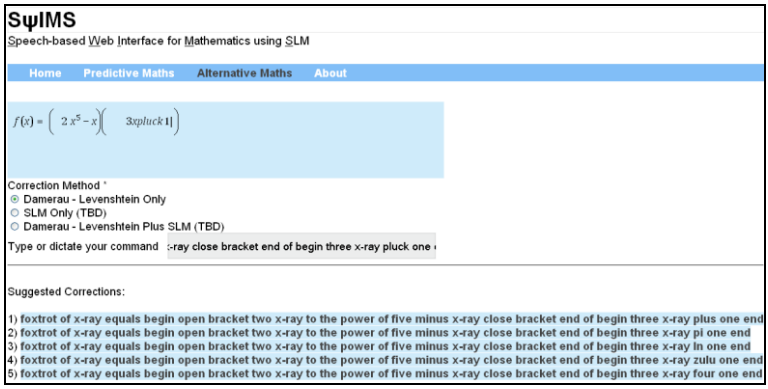

Figure 3. Alternative/Corrective Mathematics Interface in use. In the top-ranked suggestion, the OOV word "pluck" is replaced with "plus".

6. Initial Evaluation

In order to evaluate the predictive power of our statistical language models in the context of our alternative predictive mathematics system (*SWIMS*), we set up three experiments. From our previous studies of perplexities [8], we demonstrated that such

SLMs have the potential to be useful for prediction of mathematical text. The current study empirically evaluates these models when put into practice. For the first two experiments, A1 and A2, we trained a SLM using 90% of our database of spoken mathematical equations (≈3700 expressions). The remaining 10% (≈400) was then used to test the predictions offered by the system, based on the trained model, comparing these with the complete version of each expression. For the third experiment, A3, we varied the size of the training and test data sets in order to monitor the consequential change in the system's prediction performance. In experiment A4, the correction algorithm of the *SWIMS* system has been evaluated by artificially introducing a controlled selection of mistakes into otherwise "correct" expressions.

6.1. Experiments A1 and A2- prediction success rate depends on number of alternatives

Each expression in the test set was run through the interface with the last one (Experiment A1) or two (Experiment A2) word(s) omitted. We then observed the next word(s) predicted by the system, to see if one of highest ranked predictions (by probability) contained the actual missing word(s). The word "end" is normally used as a "context cue" within our specialized language for spoken mathematics, resulting in it being the most common word (see Table 1). Hence, we did not test expressions ending with "end" (A1) or ones which had "end" in the last two words (A2). Table 2 illustrates the percentages of times the correct prediction was included in the list of M "best" suggestions being offered to the user, and how this varied with M. In order to check that the results obtained were consistent, we performed 10 fold cross validation by dividing the complete dataset into 10 "folds", using one fold as the test set whilst the other 9 folds were used to build the SLM in each trial. The results of Experiment A1 are also represented graphically in Figure 4, which shows that the success of the one word ahead prediction increased as the number of suggestions shown to the user was increased, but with diminishing return.

Table 2. Experiment A1: Variation of success rate of one word ahead prediction with number of suggestions offered to the user.

	Size of Training Set	Size of Test Set	Number of Suggestions				
			5	10	15	20	25
Min	3684	407	62%	74%	79%	85%	88%
Mean	3691.8	410.2	63.2%	77.6%	84.4%	88.9%	91.1%
Max	3695	418	66%	81%	87%	91%	94%

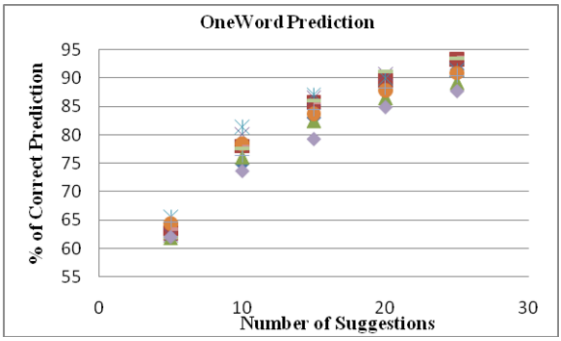

Figure 4: One word ahead prediction success rate increasing with the number of suggestions offered to the user

Experiment A2 evaluated the two words ahead prediction within *SWIMS*, in a similar manner to Experiment A1. The results are summarized in Table 3, and graphically in Figure 5. The trend is similar to that for A1, but the success rates in A2 are lower for a given number of suggestions.

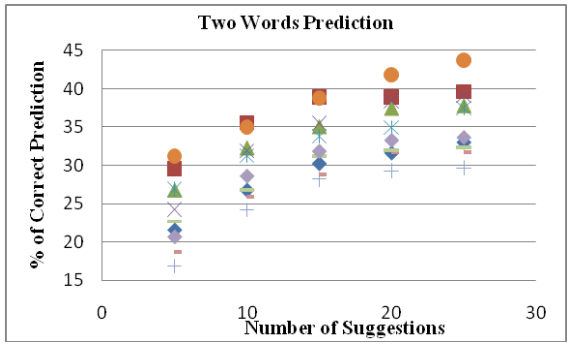

Figure 5: Two word ahead prediction success rate increasing with the number of suggestions offered to the user

Table 3. Experiment A2: Variation of success rate of two word prediction with the number of suggestions offered to the user.

	Size of Training Set	Size of Test Set	Number of Suggestions				
			5	10	15	20	25
Min	3684	407	17%	24%	28%	29%	30%
Mean	3691.8	410.2	24.3%	30.2%	33.6%	35.2%	36.2%
Max	3695	418	31%	36%	39%	42%	44%

6.2. Experiment A3- prediction success rate depends on size of training dataset

In Experiment A3, we observed how the success rate for one word prediction varied as different sized data sets were used to train the SLM. The results are summarized in Table 4.

Table 4. Experiment A3: Variation of success rate of one word ahead prediction with SLM size (5 suggestions per trial). MTrSS is Mean Training Set Size, MTSS Mean Test Set Size and M% Mean %

MTrSS	3691.8	3281.6	2871.4	2461.2	2051	1640.8	1230.6	820.4	410.2
MTSS	410.2	820.4	1230.6	1640.8	2051	2461.2	2871.4	3281.6	3691.8
M%	63.10%	62.60%	62.30%	62.10%	61.30%	60.30%	59.00%	56.50%	52.70%

6.3. Experiment A4- how successful the correction system is at correcting errors

In order to evaluate the performance of the correction algorithm, we artificially introduced some controlled errors into each of 100 expressions selected from a test expressions, then observed the proportion of these where the "correct" version was found within the 5 top ranked alternatives offered by our correction system. This was carried out for each of introducing R characters per expression, deleting R characters per expression and swapping R pairs for adjacent characters, for each of R = 1, 2, 3. The percentage of expressions which were successfully corrected using this approach for each trial are shown in Table 5. It can be seen that our method is extremely successful in correcting up to 3 insertions or transpositions of characters per expression, and fairly successful in correcting cases where up to three characters have been deleted from an expression. However, investigation of its performance in cases involving more complex or larger number of errors will require further experiments.

Table 5. Experiment A5: Variation of success rate (%) of correction using Damerau-Levenshtein method (5 suggestions offered per trial)

Number of Changes	1	2	3
Deletion of characters	95	92	68
Insertion of characters	100	98	97
Swapping pairs of adjacent characters	100	95	91

7. Discussion and Conclusion

From Experiment A1 and A2, we observe that one word ahead and two word ahead prediction success rates can be improved by increasing the number of alternatives, M, suggested to the user. However, the rate of increase of success rate diminishes as M increases, and it would appear that the maximum possible rates are about 90% for one word prediction, but around just 40% for two word prediction. However, if the user has to read too large a number of suggestions, the cognitive load imposed will very great. Thus, the number of options displayed must be limited. Based on our results, we propose that between 5 to 10 suggestions should be offered for one word prediction, giving success rates from 63 to 80%. Displaying the alternative expressions rendered into standard mathematical notation may help ease the reading burden on the user. However, two word prediction is rather less useful unless a large number of suggestions is presented. Experiment A3 showed that a small increase in success rate for one word ahead prediction could be achieved by increasing the amount of training data used, whilst M remained fixed. This is consistent with other studies of the predictive power of models based on other types of text [4, 22 and 23]. Finally, according to the results we obtained in Experiment A4, the Damerau-Levenshtein based correcting method is highly successful at correcting up to 3 errors at the character level errors in an expression. However, when the number or complexity of such errors is increased, the efficiency of correction declines.

At present, the predictions are limited to the vocabulary of the SLM. This implies that each time a new word (in our case, a spoken name of a mathematical entity) is encountered it will need to be added to the system's vocabulary. This should be relatively straightforward for the Damerau-Levenshtein based method. However, in order to modify the SLM, the corpus of mathematical expressions will have to be extended to reflect the change. Although possible in principle, this is not straightforward as one would have to find a considerable additional amount of data in order to update the model. Online learning within an adaptive system may be the solution to this issue.

Our work to date has indicated that the prediction/correction assistive facilities incorporated into *SWIMS* have potential to help make mathematical editing systems, including *TalkMaths*, more powerful and user-friendly. Improving such systems in this manner should in turn make writing and editing mathematics in electronic documents much easier for three groups – the disabled, on-line (particularly "at a distance") learners and people relying heavily on the use of portable devices – for whom these tasks are currently very difficult or even near impossible.

8. Future Work

Our method should be easily adaptable to use in several other closely related domains. For example, computer algebra systems such as Maple and Mathematica have their own language and syntax for mathematical expressions. It should be fairly straightforward for our predictive and corrective system to be integrated with these systems, assuming that enough data on past usage is available to train the models.

We have noted the issue of the high cognitive load on the user when having to read a large number of possible alternatives offered by the system. We aim to provide previews each of these, rendered into standard mathematical format, which should be make the user's task of identifying the correct version easier.

The corrective method we have used is currently not based on probabilities given by SLMs. Our next goal is to make use of SLMs to further enhance the current

Damerau-Levenshtein based correcting method, re-ordering candidate expressions by "likelihood". We also consider possibilities of incorporating a user's own "common usage" statistics, or specialized topic dependent models, in addition to our baseline SLM, so that an adaptive approach can be implemented into our system, fine-tuning it to be appropriate to the current need. Previous authors have tried to improve on N-gram models by combining these with "cache" or "word trigger" models [18, 45] in an attempt to incorporate longer-range statistical relationships between words. These approaches may prove fruitful for our system. We also aim to investigate ways we can adapt the Damerau-Levenshtein algorithm to correct speech recognition errors by applying a phonetically-weighted version to the phonetic transcription of the recognized words. For example, we would expect a higher rate of confusion between two similar phonemes, such as voiceless fricatives /s/ ("s") and /ʃ/ ("sh"), or between voiced plosives /d/ and /g/, than between two phonemes of different types [35].

References

[1] F. Damerau, A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 1964, Vol. 7, Issue 3, pp. 659-664.

[2] V. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 1966, Vol. 10, Issue 8, pp. 707-710.

[3] D.R. Attanayake, E. Pfluegel, J. C. W. Denholm-Price, G. J. A. Hunter, Architectures for Speech-Based Web Applications, *Proc. 4th International Conference on Semantic E-business and Enterprise Computing (SEEC2011)*, July 20-22, U.K. 2011

[4] A. Wigmore, Speech-Based Creation and Editing of Mathematical Content. Ph.D. Thesis, Kingston University, U.K., 2011

[5] A. I. Karshmer, Access to mathematics and science, *Proc. 11th International Conference on Computers Helping People with Special Needs (ICCHP)*, 2008, Vol. 5105, pp. 873–874.

[6] TalkMaths Project, http://www.TalkMaths.org

[7] P. Clarkson, R. Rosenfeld, Statistical Language Modeling using the CMU-Cambridge Toolkit , *Proc. Eurospeech* 1997, Vol. 5, 2707-2710. Toolkit available on-line http://www.speech.cs.cmu.edu/SLM_info.html

[8] D.R. Attanayake, G. J. A. Hunter, J. C. W. Denholm-Price, E. Pfluegel, Interactive error correction using statistical language models in a client-server interface for editing mathematical text, *Designing Inclusive Systems – Designing Inclusion for Real-World Applications*, Ed. P Langdon et al. Springer Verlag, London, Chapter 13, pp. 125-132.

[9] S. Young, Large Vocabulary Speech Recognition : A review, *IEEE Signal Processing Magazine*, 1996, Vol. 13, Issue. 5, pp. 1-4.

[10] S. Young, Talking to Machines – Statistically Speeching, *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Denver, Colorado, U.S.A, 2002

[11] Google, Inside Google Translate, 2011, http://translate.google.com/about/index.html

[12] G. Law, Phonetic alphabets (alpha bravo charlie delta), 2007 http://www.faqs.org/faqs/radio/phonetic-alph/full/

[13] I.J. Good, The population frequencies of species and the estimation of population parameters, 1953, *Biometrika*, Vol. 40, (3–4), pp. 237–264

[14] R. Fateman. How can we speak math? www.cs.berkeley.edu/~fateman/papers/speakmath.pdf, 2012.

[15] A. Wigmore, G. Hunter, E. Pfluegel, J. Denholm-Price, M. Colbert TalkMaths Better ! Evaluating and improving an intelligent interface for creating and editing mathematical text. *Proceedings of 6th International Conference on Intelligent Environments*, July 2010, Kuala Lumpur, Malaysia.

[16] L. A. Chang, Handbook for spoken mathematics (Larry's speakeasy). Lawrence Livermore National Laboratory, University of California, USA, 1983.

[17] S. Katz, Estimation of probabilities from sparse data for the language model component of a speech recognizer, *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1987, Vol. 35, pp. 400-401

[18] R. Rosenfeld, Two decades of statistical language modeling: where do we go from here?, *Proc. of the IEEE*, 2000, Vol. 88 (8) (August), pp. 1270 – 1278

[19] L.R. Bahl, F. Jelinek & R.L.Mercer, A maximum likelihood approach to continuous speech recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 1983.

[20] R. Kneser and H. Ney, Improved backing-off for m-gram language modeling, *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP),* vol. 1, pp. 181 - 184 , 1995

[21] F. Jelinek, Up from trigrams ! - the struggle for improved language models, *Proceedings of EuroSpeech*, 1991

[22] R.K. Moore, There's No Data Like More Data - but when will enough be enough ? Proceedings of the Institute of Acoustics, 2001, Vol. 23 (3), pp. 19-26

[23] G. Hunter, M. Huckvale, Is it Appropriate to Model Dialogue in the Same Way as Text? A Comparative Study Using the British National Corpus, *Proc. 2006 European Modelling Symposium*, London, U.K, pp. 199-203

[24] T. Sancho-Vinuesa et al, Automatic verbalization of mathematical formulae for web-based learning resources in an on-line environment*, INTED2009 Proceedings*, 2009, Valencia, Spain, pp. 4312–4321

[25] J. Cuartero-Olivera et al, Reading and writing mathematical notation in e-learning environments, 2012, eLC Research Paper Series, Universitat Oberta de Catalunya, Spain, http://elcrps.uoc.edu/ojs/index.php/elcrps/about

[26] Y. Solomon & J. O'Neill, Mathematics and Narrative, *Language and Education*, 1998, Vol. 12 (3), pp. 210-221

[27] M. Österholm, Characterizing Reading Comprehension of Mathematical Texts*, Educational Studies in Mathematics*, 2006, Vol. 63 (3), pp. 325-346

[28] P. Gaura, REMathEx - Reader and Editor of the Mathematical Expressions for Blind Students, *Proceedings of 8th ICCHP 2002*, Springer-Verlag, London, U.K, pp. 486-493.

[29] N. Soiffer, MathPlayer: web-based math accessibility. In Proc. of the 7th int. ACM SIGACCESS conf. on Computers and Accessibility (Assets '05), 2005, ACM, New York, NY, USA, pp. 204-205

[30] C. Bernareggi & V. Brigatti, Writing mathematics by speech: A case study for visually impaired, *Proceedings of 11th ICCHP 2008*, pp. 879–882

[31] T. V. Raman, Audio system for technical readings, 1998, Springer Verlag, Berlin

[32] Metroplex Voice Computing, Inc. mathtalk.com, http://www.mathtalk.com/

[33] C. Guy, M. Jurka, S. Stanek, and R. Fateman, Math speak & write, a computer program to read and hear mathematical input, 2004, Technical report, University of California, Berkeley, Electrical Engineering and Computer Sciences Department

[34] T. Hanakovic & M. Nagy, Speech recognition helps visually impaired people writing mathematical formulas, Proceedings of 10th ICCHP 2006, pp. 1231–1234.

[35] P. Ladefoged & K. Johnson*, A Course in Phonetics* (Sixth Edition), Cengage Learning, 2010

[36] H. Ferreira, D. Freitas, Enhancing the Accessibility of Mathematics for Blind People: The AudioMath Project, 2004, K. Miesenberger et al (eds.) *Proc. ICCHP 2004*, Springer LNCS, vol. 3118, pp. 678–685.

[37] Math in Braille Project, 2011, http://www.mathinbraille.at/en

[38] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, & T. Kanahori, Infty - an integrated OCR system for mathematical documents. In C. Vanoirbeek, C. Roisin, E. Munson (eds.) *Proceedings of ACM Symposium on Document Engineering*, 2003, pp. 95–104, http://www.inftyproject.org/en/index.html

[39] S. M. Watt and X. Xie, Recognition for Large Sets of Handwritten Mathematical Symbols , *Proc. IEEE Int. Conf. on Document Analysis and Recognition, (ICDAR 2005)*, Korea, IEEE Press, pp. 740-744

[40] A. Fujiyoshi, M. Suzuki, S. Uchida, Syntactic Detection and Correction of Misrecognitions in Mathematical OCR, *Proceedings of The 10th International Conference on Document Analysis and Recognition, ICDAR 2009*, Barcelona, Spain, pp.1360-1364

[41] E. Smirnova & S.M. Watt, Context-Sensitive Mathematical Character Recognition, *Proc. IAPR Int. Conf. on Frontiers in Handwriting Recognition, (ICFHR 2008),* August 19-21 2008, Montreal, Canada,

[42] N. Jacobs, "MathGenie : User's Guide and Teacher's Manual", 2006, http://logicalsoft.net/MathGenie.pdf

[43] K. Vertanen & D.J.C Mackay, Speech Dasher: Fast writing using speech and gaze. *Proceedings of CHI* 2010, U.S.A., pp. 595-598.

[44] P. Zielinski, Opengazer: opensource gaze tracker for ordinary webcams, 2007, http://www.inference.phy.cam.ac.uk/opengazer/

[45] R. Kuhn & R. D. Mori, A cache-based natural language model for speech reproduction, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 1990, Vol.12(6), pp. 570–583