Predictive Input Interface of Mathematical Formulas

Yoshinori Hijikata, Keisuke Horie, Shogo Nishida

▶ **To cite this version:**

Yoshinori Hijikata, Keisuke Horie, Shogo Nishida. Predictive Input Interface of Mathematical Formulas. 14th International Conference on Human-Computer Interaction (INTERACT), Sep 2013, Cape Town, South Africa. pp.383-400, 10.1007/978-3-642-40483-2_27 . hal-01497449

HAL Id: hal-01497449
https://inria.hal.science/hal-01497449

Submitted on 28 Mar 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Predictive Input Interface of Mathematical Formulas

Yoshinori Hijikata, Keisuke Horie, Shogo Nishida[1]

[1] Graduate School of Engineering Science, Osaka University
Toyonaka, Osaka (Japan)

hijikata@sys.es.osaka-u.ac.jp, horie@nishilab.sys.es.osaka-u.ac.jp, nishida@sys.es.osaka-u.ac.jp

Abstract. Currently, inputting mathematical formulas into a document using a PC requires more effort by users than inputting normal text. This fact inhibits the spreading of mathematical formulas as internet contents. We propose a method for predicting user's inputs of mathematical formulas using an N-gram model: a popular probabilistic language model in natural language processing. Mathematical formulas are usually presented in hierarchical structure. Therefore, our method incorporates hierarchical information of mathematical formulas to create a prediction model. We try to achieve high prediction accuracy of inputting characters for mathematical formulas.
Keywords: mathematical input; probabilistic language model; predictive input; N-gram model.

1 Introduction

Mathematical formulas are helpful tools for representing knowledge in all research fields such as science, engineering, social science, and economics. The World Wide Web Consortium (W3C) has standardized the markup language for representing mathematical formulas, called MathML. Users have become able to present mathematical formulas on web pages. Most web browsers and formula manipulation software comply with MathML standards. Community sites related to mathematics are just beginning to become widespread. Those users share how to solve mathematical problems and teach each other their knowledge of mathematics. In the internet research field, search technologies are beginning to be applied to searching for mathematical formulas on web pages [4, 16]. It is believed that potential needs for presenting mathematical formulas are becoming greater on the web.

However, mathematical formulas have still not spread to the internet as a representation media to the same degree as other media such as text, graphics, sounds and movies. The currently available bothersome methods of inputting mathematical formulas are a deterrent against formula usage on the web. Although we usually use only a keyboard when inputting natural languages, one must use both a keyboard and a mouse when inputting mathematical formulas. Mathematical formulas are not simp-

ly presented as the sequence of numerical numbers, alphabet and other symbols. They are usually presented as a part of some structures such as fraction and exponent, which cannot be input using a keyboard. Users input them by clicking special buttons for the functions on the formula editor with a mouse. We believe that insisting on users using both a keyboard and a mouse engenders irritation when inputting mathematical formulas.

In this paper, we propose an input method incorporating a function to forecast mathematical characters that the user will input next. For predicting the subsequent characters that the user will input, the forecasting method uses the characters in the mathematical formulas that the user has already input. The target of the forecast is limited to mathematical structures or characters that cannot be input using a keyboard. Then, using keyboard, a user selects a prediction proposal given by our input method.

Predictive text entry is popular for inputting natural language, especially for handheld devices. In the research area of natural language processing, dictionaries are invariably used to realize predictive text entry [1, 5, 13]. The dictionary usually covers text elements with high appearance frequency. The forecasting method using the dictionary outputs prediction proposals when the text part the user has input most recently begins with some text in the dictionary. It shows the user the remainder of the matched text parts in a word or a phrase as a prediction proposal. It usually orders prediction proposals according to the most recently used order or the most frequently used order [3].

We need dictionaries specialized for mathematical formulas if we apply this method to mathematical formulas. However, unlike natural language, the subsequent characters are not narrowed down in mathematical formulas when a character is input. For example, "no see" usually comes after "Long time" in English. However, the connections among characters in mathematical formulas are not so definitive. It is difficult to create a dictionary that is effective for forecasting mathematical formulas. Therefore, we propose to make a prediction based on a probability model. We apply a probabilistic language model [7] that is popular in natural language processing, to mathematical formulas and to make a prediction using the probability output from this model. This method outputs prediction proposals in the order of the probability of the next input for the user.

We assume that our input method works on an ordinary formula editor such as Microsoft Office Formula Editor or MathType. These formula editors usually provide an input interface based on What You See Is What You Get (WYSIWYG)[1] . Users can check their input formulas on the screen immediately after they input them. They also provide buttons for inputting characters which cannot be input using a keyboard on the top of the screen. In this study, we implemented our original formula editor to evaluate our proposed input method. Our formula editor follows the characteristics of ordinary formula editors described above.

[1] WYSIWYG (What You See Is What You Get) is technology which provides an input interface where the content displayed on the screen matches the content of process (especially printed results).

2 Related Works

One popular input method of mathematical formulas is TeX. Users cannot see the input formula immediately after they input each character when using TeX. Formula editors are more popular for inputting mathematical formulas. Users can check the input formula directly because it complies with WYSIWYG. The formula editor in Microsoft Office, along with its enhanced version MathType and InftyEditor are popular formula editors. InftyEditor allows users to input a math structure by inputting a command. However, they must learn the commands in advance. Handwriting input has been studied for inputting formulas [8, 12, 17]. In these studies, the systems divide the input streams into a token using the user's input stroke (or gesture). Usually, they conduct character recognition by matching the extracted tokens to the characters in the database. Finally, they infer the structure of the formula using the stream of the identified characters. Although handwriting input requires a special input device like pen input, our proposed input interface does not require it.

Input word prediction has been studied since the 1980s in the field of natural language processing. Input characters have usually been predicted in a word unit [3]. Predicted words are usually provided when a user inputs a few beginning characters in the word [1, 5, 13]. Prediction is done by matching the input characters to the dictionary [9]. Especially when users input long characters, they want predictive word entry. However, the next characters to input become various in the mathematical formula. Therefore, it is difficult to apply the dictionary-based prediction for the predictive math entry. We introduce the N-gram model, a popular probabilistic language model, into predictive math entry. An N-gram model is usually used for predictive text entry in the research area of Augmentative and Alternative Communication (AAC) [14, 15]. Reactive Keyboard is a typical study of word prediction using an N-gram model for AAC [5]. In that study, a tree is built for the prediction, where one alphabetical character corresponds to a node. Priority is assigned to each node based on the number of occurrences of the N-gram. When a user inputs some characters, the method matches them with nodes in the tree, words in the children of the matched node are provided as prediction proposals. In another research area, Zweig et al. investigate methods for answering sentence completion questions using an N-gram model [18]. Although all of their methods do not consider the structure of a sentence, our method considers the structure of mathematical formulas.

In recent years, the predictive entry of natural language has been put to practical use in cell phones and smart phones. POBox is a major predictive entry system of Japanese [10]. It outputs a word that matches the starting few characters. It also outputs characters followed by the recent input words. T9 is a popular predictive entry for cell phones [6]. Characters are divided into nine groups in T9. Each group is assigned to one key in the cell phone. Matched words in the dictionary are shown to the user as prediction proposals if a user pushes a key. Predictive entry is useful for cell phones because they are equipped with limited number of buttons. Our assessment is that it is also useful for inputting mathematical formulas because current keyboards are not equipped with keys for numerous special mathematical characters.

3 Probabilistic Language Model

3.1 Introduction of a Probabilistic Language Model

The basic role of the probabilistic language model is to calculate the string generation probability $P(w_1^n)$ under a given string of words $w_1^n = w_1 \cdots w_n$. Each of $w_1, w_2, \cdots w_n$ stands for a word. $P(w_1^n)$ can be transformed to the following formula using the multiplication rule in probability theory [7].

$$P(w_1^n) = \prod_{i=1}^{n} P(w_i \mid w_1^{i-1}) \tag{1}$$

An N-gram model is a popular probabilistic language model in the natural language processing. We propose a method for predicting user's inputs of mathematical formulas using an N-gram model.

3.2 N-gram model

Generally, when the probability of an event that might occur at some point in time is influenced only by the events which happened at the last N time point, we call this phenomenon an N-th order Markov Process [7]. An N-gram model is a model that approximates word occurrences as an N-1-th order Markov Process. In other words, it is considered that the occurrence of a word at some time point depends only on the last N-1 words. The general prediction model in the N-gram model becomes the following.

$$P(w_n \mid w_1^{n-1}) = P(w_n \mid w_{n-N+1}^{n-1}) \tag{2}$$

The cases of N=1, 2, 3 are respectively called unigram, bigram and trigram. The actual formula for calculating the probability of N-gram model becomes the following.

$$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n})}{C(w_{n-N+1}^{n-1})} \tag{3}$$

We designate this probability as the N-gram probability. The calculation of this probability presents the process by which the user inputs the next word in the context that the user has input the recent words. This helps the prediction of the next input. Here, $C(w_1^n)$ stands for the number of occurrences of string of words w_1^n in the learning data.

4 Prediction method of math input

4.1 Problems in modeling math input

In predictive text entry, a user inputs characters sequentially. The system predicts a word to be input next, as inferred from the characters that the user has already input. Natural language is a simple time series of data when we specifically examine the apparent sequences of characters. Therefore, it can be modeled appropriately by the probabilistic language model explained in the preceding section. Mathematical formulas have structures in their presentation. They are not simple symbolic sequences. For example, mathematical formulas including fractions or integrals have hierarchical structures. This indicates that mathematical formulas cannot be modeled using simple probabilistic language models.

4.2 Hierarchical N-gram model

For solving the problem described above, we propose a hierarchical N-gram model. In this model, the user's log of math input (hereinafter, "log data of math input") is divided in hierarchical levels. A model is constructed in each hierarchical level.

The content and hierarchical level of the user's input are recorded in the log data of math input. Here is an example of mathematical formula for showing the actual log data.

$$\pi = \int_0^{\infty} \frac{sin^2 t}{t^2} dt \qquad (*)$$

In this formula, $\pi, =, \int, frac, d, t$ is in the first level, $0, \infty, t, sup, sin, sup, t$ is in the second level, 2 is in the third level. *frac* denotes fraction and *sup* signifies superscript. The hierarchical information is defined in advance in each character containing a hierarchical structure. For example, for the symbol of integral $\int$, it is defined that integral range exists in the lower level of $\int$. The previously defined hierarchical information helps to record logs with hierarchical levels.

For characters that can be input using a keyboard, the unit for recording the log is a variable, numerical value, operator, function like sin and log. The name of function is detected by preparing a dictionary in which popular function names are recorded in advance. For characters that cannot be input using a keyboard, the unit for recording the log is a character that is obtainable by clicking an input button in the formula editor. Characters that cannot be input using a keyboard are Greek alphabet characters, mathematical symbols such as differential ∂ and quantifier $\forall$, fraction, operators such as $\cap$ and $\subset$, large operators such as a summation symbol and integral symbol, accents such as tildes and circumflexes, script such as subscript and superscript. One of the log data of the above formula becomes the following. The number written in a parenthesis is the character's hierarchical level in the formula.

$\{\pi(1), =(1), \int(1), 0(2), \infty(2), frac(1), t(2), sup(2), 2(3),$
$sin(2), sup(2), 2(3), t(2), d(1), t(1)\}$

For modeling the math input, the log data are divided by the hierarchical level. The log data corresponding to the k-th level are designated as "k-th level log data". Not only the characters in the k-th level but also those in the higher level are used for representing the k-th level log data because, when predicting the next input in the lower level, the last characters in the upper level can be a trigger here. For example, in the sequence $\int(1), 0(2), \infty(2)$ appeared in the above log data, 0 and ∞ is often used for integral range. Therefore, these characters occurred by $\int$ as a trigger. The log data of each hierarchical level obtained from the above log data are shown below.

- 1-st level log data : $\{\pi, =, \int, frac, d, t\}$
- 2-nd level log data : $\{\pi, =, \int, 0, \infty, frac, t, sup, sin, sup, t, d, t\}$
- 3-rd level log data : $\{\pi, =, \int, 0, \infty, frac, t, sup, 2, sin, sup, 2, t, d, t\}$

The N-gram probability is calculated in each hierarchical level. The N-gram probability corresponding to the k-th level log data is designated as the "k-th level N-gram probability". The model of math input considering hierarchical level is designated as the "hierarchical N-gram model".

4.3 Prediction using hierarchical N-gram model

A learning dataset is required for constructing a hierarchical N-gram model. It is ideal to create the model from the user's own log data. However, it is difficult to prepare a massive amount of log data of a target user (a user who will use the predictive math entry) in advance. Therefore, we prepare general log data of math input obtained from several users. The N-gram probability in each hierarchical level is calculated based on the general log data. To reflect the target user's input pattern in the model, the input log obtained while the user uses the predictive math entry is added to the log data. The hierarchical N-gram model is updated using the additional log. The prediction is made based on the N-gram probability according to the hierarchical level where the user sets focus by a keyboard or mouse. Characters with high probability to the last N-1 inputs are output as prediction proposals.

The method predicts characters to be input next in the same unit used for recording the log. It does not predict the combination of several input units at one time. The occurrence patterns become increasingly diverse and the prediction accuracy might become low if we predict the occurrence of the combination of input units. Prediction proposals are limited to characters that cannot be input using a keyboard. For math input, it is easier to input characters directly using a keyboard than to input them using predictive math entry if they can be input using a keyboard.

4.4 Smoothing

Generally, an N-gram model has a problem called the zero frequency problem. Because the N-gram probability is calculated using occurrence frequency, the probability

for a word pair that does not occur in the learning dataset becomes zero. This fact suggests cases in which no prediction proposal is presented. To solve this problem, we conduct a smoothing of the probability value. We adopt a smoothing method called linear interpolation [7], which calculates the N-gram probability $P(w_n \mid w_{n-N+1}^{n-1})$ using not only the N-gram probability but also using the lower-order M-gram probability (M < N). Actually, it linearly interpolates N-gram probability $P(w_n \mid w_{n-N+1}^{n-1})$ and lower order M-gram probability in the following equation.

The N-gram probability is calculated as follows if N=2 (bigram).

$$P(w_n \mid w_{n-1}) = \lambda P(w_n \mid w_{n-1}) + (1-\lambda) P(w_n) \tag{4}$$

Therein, λ is the interpolation coefficient. We set $\lambda = 0.7$ from our experience. The N-gram probability is calculated as follows if N=3 (trigram).

$$P(w_n \mid w_{n-2} w_{n-1}) = \lambda_3 P(w_n \mid w_{n-2} w_{n-1}) + \lambda_2 P\left(w_n \mid w_{n-1}\right) + \lambda_1 P(w_n) \tag{5}$$

In that equation, λ_3, λ_2 and λ_1 respectively represent interpolation coefficients for a trigram, bigram, and unigram. We set $\lambda_3 = 0.5$, $\lambda_2 = 0.3$, $\lambda_1 = 0.2$ respectively from our experience. We used the values above for interpolation coefficients in a later experiment.

5 Interface of predictive math entry

We implemented an interface equipped with our proposed prediction method. The interface was implemented in JavaScript. It runs on major web browsers. Figure 1 portrays our developed interface, which conforms to general formula editors. Therefore, the interface realizes WYSIWYG. It has buttons for inputting characters that cannot be input using a keyboard. The usage of our interface is the following. When a user moves a cursor to the place where the user inputs characters, predictive proposals are shown below the cursor in the predictive proposal display (see Figure 1). The user uses cursor buttons of the keyboard to move the cursor. The user moves a cursor to the predictive proposal display by hitting the space key if the user wants to select a character from the predictive proposals. To select a character from the predictive proposals, the user moves the focus to the character that the user wants to input by hitting the space key consecutively or using the cursor buttons. The selected character is input in the target place in the formula if the user pushes enter key. The user can obtain the input formula as in the form of MathML code by clicking the “get MathML” button.

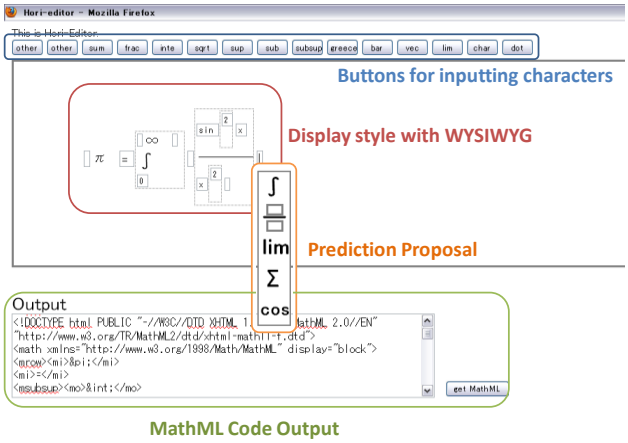

Fig. 1. Image of a mathematical editor with our prediction method of mathematical inputs

6 Evaluation of prediction accuracy

The objective of this evaluation is to ascertain (1) whether our proposed prediction method outputs more highly accurate prediction results than those provided by baseline methods, and (2) whether the hierarchical N-gram model is effective for math input. As baseline methods, we use a method using only the user's recent inputs, a method using only the user's past inputs, which means that the method uses user's recent inputs, and a method using both methods. These baseline methods can be regarded as the simplest prediction methods. To evaluate the effectiveness of hierarchical modeling, we compare our hierarchical N-gram model to a general N-gram model that uses no hierarchical information.

6.1 Evaluation Data Set

We manually selected 1,000 mathematical formulas from a textbook of mathematical analysis [11] for evaluation. We invited six test subjects to input those mathematical formulas to build a dataset. The six test subjects were divided into two groups comprising three persons each. One group inputs half of the mathematical formulas. The other group inputs the remainder. Each test subject in the same group inputs the same 500 mathematical formulas. Consequently, three pieces of log data per mathematical formula are obtained. Our dataset contains 3,000 pieces of log data of math input.

We apply ten-fold cross validation to these datasets and evaluate our proposed method in terms of its prediction accuracy. In detail, the mathematical formulas are divided into 10 subsets. Then nine subsets are used as a learning dataset, another subset is used as an evaluation dataset. To calculate the prediction accuracy, we presume

that a user inputs each formula of the evaluation dataset from its head to the tail. The prediction is made in each input position (see Figure 2). We examine which prediction proposal matches the actual character at the input position in the log data of the evaluation dataset. That is to say, at each input position in a mathematical formula in the evaluation dataset, we acquire prediction proposals using N-1 recent inputs extracted from the log data. We regard the character at each input position in the log data as a correct character and examine which prediction proposal corresponds to the correct character. We calculate prediction accuracy as a ratio of the input position where the top k candidates of the prediction proposals include the correct character in the log data to the entire input positions. We use the top 10 candidates for evaluation. This calculation is done for all input positions for all formulas in the test dataset.

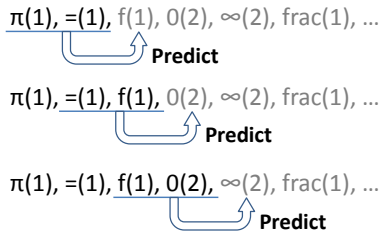

Fig. 2. Method for calculating prediction accuracy

6.2 Baseline Method

The following prediction methods are compared with our proposed prediction method. We consider these prediction methods as baselines.

- Prediction using recent inputs (Recent): This method considers the character existing at the k-th former position in the input log as a prediction proposal ranking at k-th order. It outputs prediction proposals to k=10. When k<10 at the current input, it outputs prediction proposals within the k-th rank.

- Prediction using recent inputs and their frequencies (Rct & Frq): This method orders the prediction proposals output by the recent input method described above according to their frequencies. The frequency of each candidate is calculated by counting its occurrences in the formula that the user is currently inputting.

- Unigram (N=1): This method makes a prediction based on the N-gram model (N=1). The prediction proposals become the top-k characters according to the frequencies in the whole dataset. It always outputs the same prediction proposals.

- Unigram (N=1) + Recent input method (Rct & N=1): This method is a hybrid method of unigram model and the recent input method. It calculates the appearance probability for all the characters using unigram model. For characters which exist in the former k characters in the log data of math input, it calculates the probability that a character at the j-th rank in the recent input method becomes a correct character (the prediction accuracy of the recent input method in Table 1 which can be calculated in

advance). It adds the both probabilities and uses the added values for ordering the characters.

6.3 Comparison of our Proposed Method and Baseline Methods

We show that our proposed prediction method achieves better prediction accuracy than the baseline methods. We examine our hierarchical N-gram model (N=2, 3) and their smoothing methods (hereinafter, "N-gram model (N=2, Smoothing(Smt))" and "N-gram model (N=3, Smoothing(Smt))". The implementation becomes difficult when building an N-gram model with N = 4 and higher because it requires vast memory capacity. The prediction accuracy is calculated using the mathematical formulas in the evaluation dataset. The results are presented in Table 1. Bold values represent the best prediction accuracy among the prediction methods, which is true also for the latter table. As the results show, the proposed method (N=3 with smoothing) apparently achieves the highest accuracy. Its accuracy becomes about 89% when the number of prediction proposals to show to the user is limited to the top five ranking. Its accuracy becomes about 95% when the number of prediction proposals to show to the user is limited to the top ten ranking.

We compared our proposed method (N=3) and another proposed method (N=2). When N=3, the prediction accuracies of the top three ranking and the smaller rankings (k=1,2) become higher than when N=2. The prediction accuracies of the larger rankings (the top four ranking and the larger rankings ($k = 5, 6, \cdots, 10$)) become worse because of the increase of the N-gram pairs in the learning dataset. However, when N=2, the prediction accuracies of the larger rankings become higher than when N=3. However, the prediction accuracies of the top three ranking and the smaller rankings become worse. The method can use only the latest input when N=2. Therefore, the prediction accuracies of the smaller rankings become worse than N=3. Regarding the result of N=3 with smoothing, the prediction accuracies are high both in the smaller rankings and in the larger rankings. It is apparent that the smoothing treatment compensates the above shortcomings by linear interpolation between the trigram and bigram.

Finally, we provide some insight into the results of the baseline methods. The prediction accuracy of the top five ranking is about 41% in the recent input method. Improvement of the precision is not apparent after the top five ranking. Therefore, characters that are repeated in one formula are limited to five kinds and fewer. Compared to the recent input method, in the recent and frequent input method, the accuracies become higher in the larger rankings. However, the degree of improvement is not great. Although unigram (N=1) always outputs the same prediction proposals, it achieves nearly 80% of accuracy in the top ten ranking. However, the accuracies in the smaller rankings are worse than those achieved using our proposed method. When combining unigram and the recent input method, the accuracies become better in the top three ranking and the larger rankings. However, the accuracies of the top one ranking become worse.

Table 1. Probability of the correct charcter appearing in the top k candidates

	k=1	2	3	4	5	6	7	8	9	10
Recent	0.154	0.286	0.353	0.393	0.410	0.416	0.416	0.416	0.416	0.416
Rct & Frq	0.181	0.295	0.359	0.394	0.411	0.415	0.416	0.416	0.416	0.416
N=1	0.287	0.496	0.571	0.595	0.638	0.681	0.702	0.728	0.758	0.793
Rct &N=1	0.242	0.498	0.614	0.668	0.710	0.736	0.787	0.810	0.826	0.844
N=2	0.546	0.703	0.778	0.826	0.856	0.880	0.898	0.912	0.925	0.935
N=3	0.627	0.730	0.780	0.806	0.819	0.826	0.833	0.839	0.843	0.845
N=2, Smt	0.541	0.689	0.758	0.798	0.838	0.873	0.899	0.916	0.930	0.940
N=3, Smt	**0.657**	**0.770**	**0.833**	**0.844**	**0.888**	**0.907**	**0.921**	**0.934**	**0.945**	**0.954**

6.4 Effectiveness of Hierarchical N-gram

We validate the effectiveness of introducing hierarchical information to an N-gram for predicting the math input. We compare the case which introduces hierarchical information to the model and the case which does not introduce hierarchical information to the model. The comparison is made for N-gram model (N=3, smoothing) that achieves the best accuracy in the previous subsection.

Hierarchical information works well for a case in which the user inputs characters at a shallower level after inputting characters at a deep level. We examine the prediction accuracy for this case. Those cases cover about 15% of input positions in our dataset. Table 2 shows results for those cases. They reveal that we can increase the prediction accuracy using the hierarchical information. It might decrease the overall usability if the prediction accuracy decreases for some input conditions. We infer that incorporating hierarchical information into the model is effective for predictive entry.

6.5 Effectiveness for adding the personal log data

This subsection validates the effectiveness for adding the target user's log data to the learning data. This might deal with the inconsistency problem of the input order among users. 400 log data are always used as learning data in this evaluation. One user's 500 input logs are divided in to five sets, each of which has 100 input logs. Each set is used as test data in turn. At first, 400 input logs consist of those of the other two users. As we explained, one formula has three user’s input logs. One user is randomly selected from the other two users. The selected user’s log is used here. We increase the ratio of the target user's log data to the all log data in the learning data set from 0% to 25, 50, 75, 100% and calculate the prediction accuracy. Note that we do not add the target user's log data to the original 400 learning data but replace the log data for the same formula in the original learning data set. The number of logs in the learning data stays constant. This eliminates the influence of the increase of the learning data to the prediction accuracy. Six users' logs are used for this evaluation.

The average of the six users' prediction accuracies are presented in Table 3. The results show that the prediction accuracy increases a little when the ratio of the personal logs increases. The improvement is not so large. However, we can also say that our

prediction method achieves accurate prediction even if it does not use the target user's personal logs. It is expected that our method improves the prediction accuracy when many log data are used as learning data even if they are other users' logs.

Table 2. Probability of the correct character appearing in the top k candidates :comparing the case with hierarchical information(with) and that without hierarchical information(w/o)

	k=1	2	3	4	5	6	7	8	9	10
with	**0.603**	**0.712**	**0.790**	**0.816**	**0.845**	**0.872**	**0.889**	**0.915**	**0.928**	**0.940**
w/o	0.507	0.631	0.723	0.755	0.789	0.823	0.844	0.865	0.896	0.905

Table 3. Probability of the correct character appearing in the top k candidates :changing a ratio of log data inputted by a certain user in the learning dataset

	k=1	2	3	4	5	6	7	8	9	10
0%	0.477	0.596	0.647	0.690	0.729	0.772	0.793	0.815	0.834	0.844
25%	0.476	0.597	0.647	0.688	0.728	0.772	0.794	0.818	0.832	0.844
50%	0.478	0.597	0.648	0.688	0.728	0.775	0.797	**0.822**	0.837	0.849
75%	0.482	**0.600**	0.650	0.690	0.733	0.778	**0.800**	0.820	0.839	0.851
100%	**0.484**	**0.600**	**0.653**	**0.695**	**0.738**	**0.779**	**0.800**	0.820	**0.842**	**0.853**

7 Evaluation of Usability

The previous section showed that our proposed method outperforms other prediction methods in prediction accuracy. However, that fact does not mean that our proposed method helps users to input mathematical formulas. The objective of this section is to show that our interface of predictive math entry (an input interface incorporating our proposed prediction method) helps users' actual inputs by conducting user evaluation according to the interface's usability.

7.1 Experimental Condition

The evaluation of usability is accomplished according to quantitative indices and qualitative indices. The number of times of switching between a keyboard and a mouse (hereinafter, “#switching”), input time and the number of incorrect inputs (hereinafter, “#incorrect”) are used as quantitative indices. Questionnaires related to the usability of the interface, comprising multiple-choice questions and free description, were used as qualitative indices. In this experiment, 12 graduate and undergraduate students participated. We asked them about their experiences of inputting mathematical formulas on PCs. All users answered that they had some experiences on inputting mathematical formulas on PCs. They answered that they had used general formula editors or TeX. Among the participants, three users were good at inputting formulas using TeX and were able to input most formulas without seeing the reference.

7.2 Evaluation on the Differences among Interfaces

Experimental Method

This subsection shows the effectiveness of our interface of predictive math entry. Our interface was compared to a general formula editor and an input interface using TeX. Our interface of predictive math entry used in the experiment is the interface depicted in Figure 1. We eliminated the function of our predictive entry from the above interface. That is used as a general formula editor. We originally implemented an input interface using TeX on JavaScript. The screen shot is depicted in Figure 3. A user inputs mathematical formulas using TeX command in its text area. When the user clicks the compile button, the system outputs the compiled formulas with rendering. This interface does not comply with WYSIWYG.

The users input 60 formulas (20 formulas using each interface). The number of characters in each formula is set within some range. These formulas are selected from a textbook of mathematical analysis [11]. We carefully selected formulas different from the formulas in the learning dataset. All the 3,000 formulas used in evaluation of prediction accuracy are used as the learning data for our interface. The quantitative indices are measured in each input formula. After inputting 60 formulas, users answered the questionnaires for qualitative evaluation. The question items are shown in the 1-st column of Table 4. Q1 and Q3 are provided to ascertain which interface the users can use to input formulas with a good level of comfort. We set Q2 to find the time the user felt for inputting formulas. The actual time might differ from their sensory time. Q4 is provided to elicit how fast the user can learn the input method in each interface. Q5 is provided to find out which interface the user prefers with all evaluation viewpoints considered.

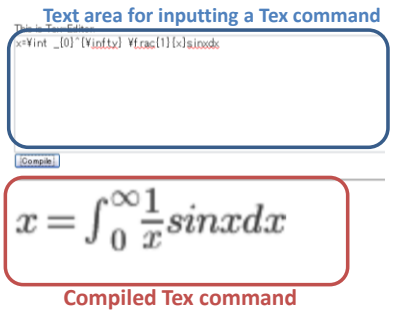

Fig. 3. Input interface using TeX

Results on Quantitative Indices

The results related to quantitative indices are presented in Table 5. The number of times necessary for key inputs and those of mouse clicks (hereinafter, #keyinputs and #clicks) are also presented in Table 5. Furthermore, we examined whether a statisti-

cally significant difference exists among the interface on these indices using a Student t-test. The results are presented in Table 6. In Tables 5 and 6, “ours” means our interface with predictive math entry, “w/o prediction” means the input interface without predictive math entry and “TeX” means the input interface using TeX. We confirmed the statistically-significant difference among interfaces in all indices except #incorrect between our interface and that without predictive entry.

The results showed on input time, it is apparent that our interface achieves the shortest input time. Input time is an important index to show that the user can input formulas smoothly. When comparing our interface and the interface without predictive entry, it is apparent that our interface decreases the #switching that is regarded as a reason that inputting mathematical formulas is bothersome. The reduction rate is about 89.1%. In the TeX interface, #incorrect becomes the three times that of the proposed interface and the interface without predictive entry. From these results, it is apparent that the incorrect inputs decrease when users can see the input formulas immediately after inputting characters. No significant difference was found between our proposed interface and the input interface without predictive math entry. However, the value of #incorrect is lower in the proposed method than the interface without predictive entry. A clear difference might be found if we increase the number of users in the experiment. #keyinputs is higher in the proposed method than the interface without predictive entry because users input characters by selecting the prediction proposal with a keyboard and move the cursor with cursor key.

Results on Qualitative Indices

The results of questionnaires for qualitative evaluation are presented in Table 4. The value for each item is the number of users who selected the item. Values shown in the parentheses are the numbers of users who can input mathematical formulas without seeing any references (hereinafter “TeX users”). Our interface achieves the best evaluation for all questionnaire items (Q1 - Q5). Especially for Q3, all users answered that they can input formulas most intuitively using our interface. In our interface, users can see the input formulas right after they input each character. Our interface also provides the prediction proposals that the users want to input. These characteristics engender the users' high evaluations to our interface. When we checked users' free descriptions, many users gave the opinion that switching a keyboard and a mouse took a burden in inputting mathematical formulas. This result supports our proposed interface that decreases the mouse input using the predictive entry.

However, some users supported the TeX interface in answers to the questionnaires. They gave the opinion that they are comfortable with the interface, and that they are used to it because they usually use TeX for inputting formulas. Actually, they were able to input formulas smoothly when using the TeX interface in the experiment. In the TeX interface, users can input formulas only by a keyboard when they learn the TeX command. Therefore, users who learn the TeX command tend to assign positive opinions to the input interface using TeX.

Table 4. Results of questionnaires for finding differences between each interfaces (The value for each item is the number of users who selected the item. The values shown in parentheses are the numbers of users who can input mathematical formulas without seeing any references)

	ours	w/o prediction	TeX
Q1. By which interface could you input the most smoothly?	10(2)	0	2(1)
Q2. By which interface could you input the most quickly?	10(2)	0	2(1)
Q3. By which interface could you input the most intuitively?	12(3)	0	0
Q4. Which interface could you get familiar with the most quickly?	11(3)	0	1
Q5. Which interface do you prefer most?	9(1)	0	3(2)

Table 5. Results of the experiment for finding differences between interfaces

	ours	w/o prediction	TeX
#clicks	1.69	13.2	0
#keyinputs	86.8	52.2	121
#switching	1.79	16.5	0
input time	66.0	74.0	89.6
#incorrect	1.38	1.57	4.43

Table 6. Results of *t*-test for the experiment for finding differences between interfaces

	Ours vs. w/o prediction	Ours vs. TeX	w/o prediction vs. TeX
#switching	***	***	***
input time	***	***	***
#incorrect		***	***

*** : $p<0.01$

7.3 Evaluation on the Differences in Prediction Accuracy

Experimental Method

This subsection presents results of an examination of the influence of the differences in prediction accuracy on usability. Prediction methods of three types were selected considering the difference in prediction accuracy. The N-gram model (N=3, smoothing) was selected as a method with high prediction accuracy. The recent input and unigram method and the recent input method were selected as a method with medium prediction accuracy and a method with low prediction accuracy, respectively. Hereinafter, we designate these methods as “high-accuracy method (high)”, “medium-accuracy method (medium)”, and “low-accuracy method (low)”. We conducted a user experiment using our input interface depicted in Figure 1. We changed the prediction method in this interface for the experiment.

The experimental method is the same as that used in the evaluation on the differences among interfaces. Only question items for the qualitative evaluation differ from those used in the previous evaluation. Table 7 presents the question items. Q3 is provided to know that the users noticed the difference in the prediction accuracy. Q1, Q2, and Q4 are the same questions as those used in the prior evaluation.

Table 7. Results of questionnaires for finding differences between different prediction methods

	high	medium	low
Q1. By which interface could you input the most smoothly?	11	0	1
Q2. By which interface could you input the most quickly?	12	0	0
Q3. Which interface's prediction did you feel the most accurately?	11	0	1
Q4. Which interface do you prefer most?	12	0	0

Table 8. Results of the experiment for finding differences between different prediction methods

	high	medium	Low
#clicks	1.32	4.00	6.99
#keyinputs	87.5	96.3	73.4
#switching	1.46	4.20	8.70
input time	54.3	62.7	61.9
#incorrect	1.21	1.43	1.53

Table 9. Results of *t*-test for the experiment for finding differences between different prediction methods

	high vs. medium	high vs. low	medium vs. low
#switching	***	***	***
input time	***	***	
#incorrect	**	*	

*** : $p<0.01$, ** : $p<0.05$, * :$p< 0.1$

Results on Quantitative Indices

Table 8 presents the results on quantitative evaluation. #keyinputs and #clicks are also shown in Table 8. We conducted Student t-test to assess differences among the prediction methods. The results are presented in Table 9. We confirmed statistically significant difference among the methods in all indices except input time and #incorrect between the medium-accuracy method and the low-accuracy method. When particularly addressing #switching in Table 8, #switching decreases when the prediction accuracy becomes high. #incorrect decreases in the high-accuracy method compared to the other methods. The input time becomes the shortest in the high-accuracy method. From these results, it is apparent that prediction accuracy influences the usability for inputting formulas.

It is particularly interesting that the input time becomes shorter in the low-accuracy method than in the medium-accuracy method. The reason for this result is the following. The low-accuracy method (recent input method) shows only those characters which the user has input before as prediction proposals. In this case, the user can expect what characters are given as prediction proposals while inputting characters. We think that users use predictive entry by considering what characters are given as prediction proposals next. In fact, when we observed the users' input activities when they used the N-gram model (N=3, smoothing), we found that some users moved the input cursor to the predictive proposal display before checking what predictive proposals are shown there. In N-gram model (N=3, smoothing), the users input formulas under the expectation in which their target character exists in the higher rank in the predictive proposal list because the prediction accuracy is highly sufficient. Actually, they input formulas very smoothly by forecasting the prediction results. With the recent input and unigram method, the users had difficulty forecasting the next prediction proposals because unigram does not achieve the high-accuracy prediction and the prediction proposals change according to the most recent inputs. Based on this result, it is important for users to forecast the next prediction proposals for predictive entry.

Results on Qualitative Indices

The results of questionnaires for qualitative evaluation are presented in Table 7. Most users selected the high-accuracy method as the best method for all question items. This result corresponds to the results on the quantitative evaluation. For Q1 and Q3, one user selected the low-accuracy method as the best method. The reason is related to the user's input activities explained in the previous subsection. The user preferred the prediction that is easily forecasted before shifting the input cursor to the prediction proposal display.

8 Conclusions

As described in this paper, we proposed an input method that predicts the next input characters for mathematical formulas. N-gram model is applied to our method, which is a popular probabilistic language model. We incorporated hierarchical information in mathematical formulas into an N-gram model. The calculated probability was used for predictive math entry. The proposed method is evaluated using prediction accuracy. Results showed that the prediction accuracy of our method is higher than that of other baseline methods. The proposed input interface was evaluated using a user experiment. Results showed that our interface outperforms the input method without predictive entry and the input method using Tex in the usability. We expect that our interface for predictive math entry shall contribute to the spread of math contents. The ease with which users can forecast the prediction proposals is important for usability. We will examine the relation between user predictability and usability for the predictive math entry.

References

1. Copestake, A. Augmented and alternative nlp techniques for augmentative and alternative communication. In *Proceedings of the ACL workshop on Natural Language Processing for Communication Aids* (1997), 37–42.
2. Darragh, J., W. H. J. L. The reactive keyboard: A predictive typing aid. *IEEE Computer 23* (1990), 41–49.
3. Garay-Vitoria, N., and Abascal, J. Text prediction systems: a survey. *Univers. Access Inf. Soc. 4*, 3 (2006), 188–203.
4. Hashimoto, H., Hijikata, Y., and Nishida, S. Incorporating breadth first search for indexing mathml objects. In *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on* (2008), 3519–3523.
5. Hunnicutt, S. Input and output alternative in word prediction. *STL/QPRS* (1987), 15–29.
6. King, M. T., Grover, D. L., Kushler, C. A., and Grunbock, C. A. Reduced keyboard disambiguating system patent, 1998.
7. Kita, K. *Gengo to Keisan – Kakuritsuteki Gengo Model (Language and Computation - Probabilistic Language Model)*. University of Tokyo Press, Tokyo, Japan, 1999.
8. LaViola Jr., J. J., and Zeleznik, R. C. Mathpad2: a system for the creation and exploration of mathematical sketches. *ACM Trans. Graph. 23*, 3 (2004), 432–440.
9. Mackenzie, I. S., and Felzer, T. Sak: Scanning ambiguous keyboard for efficient one-key text entry. *ACM Trans. Comput.-Hum. Interact. 17*, 3 (July 2010), 11:1–11:39.
10. Masui, T. Pobox: An efficient text input method for handheld and ubiquitous computers. In *Proceedings of the International Symposium on Handheld and Ubiquitous Computing - HUC '99*, Springer Verlag (1999), 289–300.
11. Miyake, T. *Introduction of Derivation and Integration*. Baihukan, Tokyo, Japan, 1992.
12. O'Connell, T., Li, C., Miller, T. S., Zeleznik, R. C., and LaViola, Jr., J. J. A usability evaluation of algosketch: a pen-based application for mathematics. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '09, ACM (New York, NY, USA, 2009), 149–157.
13. Swiffin, A., Arnott, J., Pickering, J., and Newell, A. Adaptive and predictive techniques in a communication prosthesis. In *Augmentative Aleternative Communication* (1998), 181–191.
14. Vertanen, K., and Kristensson, P. O. The imagination of crowds: conversational aac language modeling using crowdsourcing and large data sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, Association for Computational Linguistics (Stroudsburg, PA, USA, 2011), 700–711.
15. Wandmacher, T., Antoine, J.-Y., and Poirier, F. Sibylle: a system for alternative communication adapting to the context and its user. In *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, Assets '07, ACM (New York, NY, USA, 2007), 203–210.
16. Zanibbi, R., and Blostein, D. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)* (2011), 1–27.
17. Zeleznik, R., Bragdon, A., Adeputra, F., and Ko, H.-S. Hands-on math: a page-based multi-touch and pen desktop for technical work and problem solving. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, ACM (New York, NY, USA, 2010), 17–26.
18. Zweig, G., Platt, J. C., Meek, C., Burges, C. J. C., Yessenalina, A., and Liu, Q. Computational approaches to sentence completion. In *ACL (1)*, The Association for Computer Linguistics (2012), 601–610.