

Intelligent Predictive Text Input System using Japanese Language

Nwanua Elumeze, Keisuke Nishimoto

Department of Computer Science

University of Colorado at Boulder

{nwanua.elumeze, keisuke.nishimoto}@colorado.edu

Abstract

This paper proposes a new predictive text input system for Japanese language. As the user types in text, the system constructs a matrix containing word frequencies given preceding words in a sentence. It calculates the frequencies of candidate words from the matrix based on a partial word and its context, and on a list of critical words included in the partial sentence that the user is currently typing, and proposes the most likely words, sorted in order of frequency.

We evaluated the system by constructing a matrix from the training texts and counting the number of keystrokes needed to type a test text with this system. We found that it did reduce the number of key strokes required to type test sentences compared with a non-predictive and with a predictive text input system. However the reduction is also heavily dependent on word reuse in the corpus and new text.

1 Introduction

Predictive text input systems enable users to input text with fewer keystrokes by predicting the word the user is trying to type based on the user's previous input and/or the user's partial input of the intended word. A predictive text input system does not necessarily reduce the time to type in text, even if it reduces the total number of keystrokes, because it requires the user to select the correct word from multiple candidates predicted by the system (Copestake, 1997). In the case of typing in a language that does not require text conversion, such as English, with a full-sized keyboard,

predictive text input ends up not being as efficient as simply typing without prediction, because predictive text input increases user cognitive load incurred by interruption necessitated by word selection.

However, not all languages can be input without interruption by simply typing on a keyboard. In the case of Japanese three types of characters (hiragana, katakana, kanji), which contain more than 3,000 characters in total, can be used in a text. Therefore, it is almost infeasible to directly type these characters without conversion. Typical Japanese text input is performed in the following manner: first, a user types in a text in hiragana (phonetic alphabet) using an alphabetic keyboard. After typing the entire or partial sentence, the user converts the text including kanji (Chinese characters). Since Japanese has a large number of homonyms with different kanji notations, the user has to choose the correct one depending on its context, if the system's choice is wrong. Figure 1 shows an example of this process. Since interruption and selection of candidates are necessary anyway, using predictive text input would be more feasible as a way to input Japanese texts.

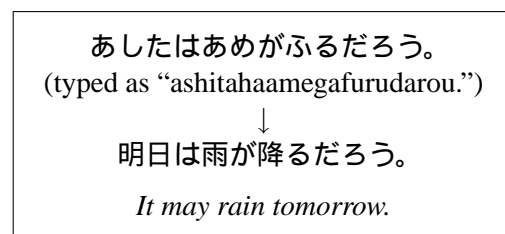


Figure 1: An example of Japanese text input

Another situation where predictive text input may be useful is typing texts on small devices, such as PDAs and cellular phones. These devices

are not usually equipped with full-size keyboards, and the user of these devices has to input text either by using pen stroke or a telephone 9-button keypad, which are much less efficient than full-size keyboards. In case of 9-button keypads, up to 4 letters are assigned to one button, requiring at worst 4 button presses to type just one letter. As text messaging on cellular phones becomes more popular and the necessity to type text on a cellular phone increases, a few predictive text input methods are proposed and implemented in commercial products. The next section introduces some of them.

2 Prior Work

POBox (Masui, 1998; Masui, 1999) is a predictive text input method originally designed for pen-based devices. POBox performs ambiguous search to predict candidate words from a user's partial input. For example, it is possible to predict a word "design" from an incomplete input such as "dsn." POBox also has a context dictionary that keeps track of which words are likely to appear before a certain word and uses it to predict words based on the previous word. Since POBox relies on a simple bigram model, it is supposed to work for many languages if a dictionary for the language is available. POBox has been implemented in several commercial cellular phones for the Japanese market.

Another predictive text input system widely used in cellular phones is T9 (Tegic Communications, Inc., 2000) (<http://www.t9.com/>) developed by Tegic Communications, Inc. It assumes a standard 9-button telephone keypad where 3 to 4 alphabets are assigned to each button. Unlike the conventional input method, the user does not need to push a button multiple times to enter a letter. Instead, the user enters a word by pushing a button per letter and T9 predicts candidate words from the button sequence. For example, if the user types a sequence of 234, there exist 27 possible words for it since 3 alphabets are assigned to each key. However, only a few of them give a correct word and can be a candidate. It seems that T9 further disambiguates words by a certain metric, but the algorithm is proprietary (Soukoreff and MacKenzie, 2003). Currently, T9 is currently available in more than 40 languages, including Japanese.

Ichimura et al. developed a Japanese text input system with prediction (Ichimura et al., 2000).

The system is dedicated for Japanese text input and employs a more complex model to increase the accuracy of prediction rather than reduce the number of keystrokes assuming a full keyboard. It uses two factors to limit the number of candidates to be predicted: a certainty factor indicating how certain a candidate is - calculated from the frequency in the corpus given a typed text, and a usefulness factor indicating how useful a candidate is - calculated as the difference between the lengths of a candidate word and of matching substrings of the typed text.

On the other hand, TouchMeKey4 keypad (Tanaka-Ishii et al., 2002) was designed for more limited input devices with just 4 buttons, as the name suggests. Similar to text input on cellular phones, the system assigns six or seven alphabet letters to a button. TouchMeKey4 uses PPM (prediction by partial match) to determine candidate words, and the relevance of each candidate is measured by the base dictionary, the unigram statistics collected from a corpus, and the user corpus, the n-gram statistics constructed from a small document written by the user.

One of the latest research on the predict text input was presented in (Komatsu et al., 2005). Their strategy was to store all the documents visited by a user to construct a corpus and utilize it to predict a word, especially long and context-specific compound words such as "Software Engineering" or "Unix programming." They developed the system by integrating Kukura, a document storage system, together with PRIME, a predictive input system. Their study revealed that some forms of text, such as a chat log, were highly context specific and the ratio of word reuse was sometimes as high as 50%, showing that this approach was feasible. We utilized this hybrid implementation strategy in our system.

3 Using a Context for Prediction

3.1 Approach

Our system was designed for inputting Japanese texts on mainly, but not limited to, small devices such as cellular phones. Our approach is to increase the accuracy of prediction by POBox by utilizing more detailed context information and knowledge of Japanese grammar.

Like POBox, our system searches for candidate words based on a user's partial input word. Whereas POBox uses just the previous word as

a context to make better predictions, our system uses an entire partial sentence to limit and prioritize candidate words. For example, if the sentence a user is typing includes the word *reluctantly* and she typed in the letter *p*, our system might infer that the word the user wants to type is likely to be *pay*, or something the user is not willing to do. This approach is especially useful in Japanese because almost all Japanese sentences end with a verb block, which is likely be restrained by preceding subject/object noun and adverb words (Figure 2). In the first example sentence in the figure, the subject pronoun 彼 (he) and the adverb しぶしぶ (reluctantly) suggests the verb block to be an action that can unwillingly be performed by a person, and the object noun 問い (question) requests the verb block to be something performed on a question.

Another benefit of using an entire partial sentence as a context is that it is more resilient to word order changes. In the case of Japanese, word order is relatively free compared with English. For example, the two sentences in Figure 2 have the same structure, but the order of blocks before the verb block are different. In this situation, a simple bigram model is less useful as a clue for prediction because one bigram sequence extracted from a sentence cannot be used for another sentence having the same set of words but with different word order. On the other hand, our model is not affected by the order of preceding words and thus is expected to give more robust prediction in Japanese text input.

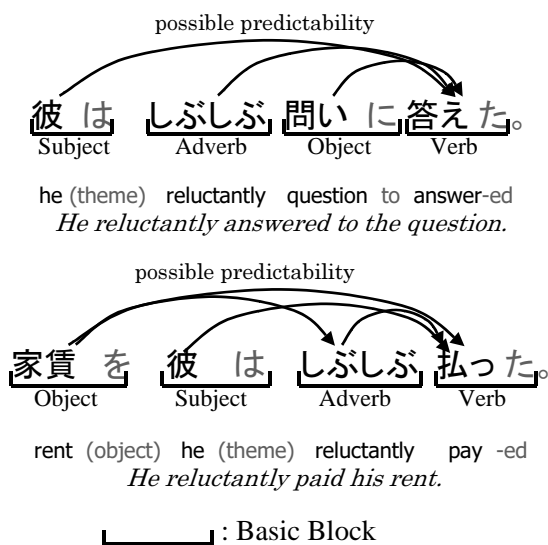


Figure 2: Possible predictability in a sentence

To further increase the accuracy of our model, we make a distinction between critical and non-critical words. To make the distinction simple, we regard the structure of Japanese sentences as a sequence of basic blocks in the above explanation. A basic block in Japanese language consists of one independent word (jiritsugo, 自立語) depicted in black in Figure 2, and zero or more ancillary words (fuzokugo, 付属語) depicted in gray. Ancillary words are either particles or auxiliary verbs, both of which are closed class words, and they never come to the start of a basic block. The Part-of-Speech (PoS) of words followed by an ancillary word is limited to noun, verb, adjective and auxiliary verb. As such, ancillary words are less important for prediction in our model than independent words.

Based on this grammatical nature of Japanese language, our model handles only independent words. When the system constructs a context from a user's previous input it extracts only independent words, and this model is used to predict only independent words. In reality, however, the system also needs to support the input of ancillary words. Our solution to this problem was to build a hybrid system that uses both our model and the bigram model and to put priority on either of them based on the present context. Since ancillary words appear only after a word with a predefined set of PoS, we put priority on the bigram model if the PoS of the previous word is included in the set. Even if a user wants to input an independent word after such words, putting priority on the bigram model does not pose much of a problem, because ancillary words are in general shorter than independent words and can be easily eliminated from the list of candidate words as the user types in more letters.

3.2 Architecture

As shown in Figure 3, our solution relies on a number of existing programs that help with prediction, conversion, and user interaction. Per the diagram, the user interacts directly with the Emacs text editor, and types a special control sequence to activate our Japanese text prediction feature. Communication between Emacs, our predictor, and the POBox predictor server are done via TCP sockets, so ostensibly, the system could serve multiple Emacs clients at once.

As the user types, Emacs sends the alphabetic characters over to our system, where possible

Japanese characters are predicted and compiled, and eventually sent back to the user via Emacs. She can then press the "space" key to scroll among the possible choices, and then types "enter" to make a selection, or continues typing more alphabet characters to further disambiguate the candidate words. The screenshot in Figure 6 illustrates this process from the user's point of view.

For a given (partial) word or context sentence, the system uses the MeCab (Japanese structure analyzer, <http://mecab.sourceforge.jp>) program to tag words as nominals, verbs, auxiliaries, etc. This information is used to capture some meaning about the structure of a given sentence as it's being typed, and it helps the system make better predictions that are influenced by the *type* of an immediately preceding word.

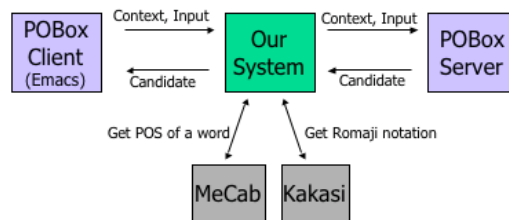
Note that a user is most likely going to use an English keyboard, and we use Kakasi (Kanji-Kana inverter, <http://kakasi.namazu.org>) to convert Japanese text into the alphabetic description of the Japanese pronunciation - this is the process of romanization. For example, the character が sounds like "ga" when spoken, so its romanized version would be the letter 'g' followed by the letter 'a'. This enables the system to determine, in real time, whether a given alphabetic string can yield certain Japanese characters (based on how the characters may be pronounced). Communication between our predictor, Mecab, and Kakasi is done via Unix pipes.

When our predictor has been trained on a corpus, the POBox sever acts mainly as a backup that offers predictions in addition to the ones we make. However, in the case of an untrained system, the POBox server becomes the primary source of predictions, and our system learns by observing the choices a user makes, as well as the contexts surrounding those choices.

When the user makes a choice from the list of candidates, Emacs sends our system the number of the selection (e.g. 84 is the code to indicate that the fourth candidate was chosen). Upon receipt, our system updates the matrix and the current context to reflect this fact.

3.3 Algorithm

The predictions our system makes depend entirely on the frequency of a word given the occurrence of preceding words in a sentence context. Typically, only the words in a given sentence are considered



MeCab: Japanese POS tagger
Kakasi: Kanji-Romaji conversion tool

Figure 3: System Architecture

彼は問いにしぶしぶ答えた
しぶしぶ彼は家賃を払った

Figure 4: New text without an existing corpus

part of a particular word's context, and the frequency with which it appears with the other words in different contexts (sentences) will indicate the likelihood of it appearing in this new context. This information is stored in a sparse matrix that essentially predicts how likely it is for a word (starting with a certain combination of alphabetic characters) to show up at particular parts of some text. Concretely, matrix[‘今日’, ‘寒’] denotes the frequency of ‘寒’ given ‘今日’. From this, given the present context, it goes on to calculate the sum of frequencies for each candidate, and returns a list of the most likely words.

To illustrate, starting with a fresh slate, the example sentences in Figure 4 were typed using Emacs, and our system immediately updates its matrix to reflect the structure in Table 1.

The two entries with the number "1" in the first column, for instance, indicate the frequency with which "彼" and "しぶしぶ" were the first words in the context. The remaining zeros in that first column indicate that no other word started a sentence, per this example. The entries with the number "1" in the second column indicate how of-

	-	彼	しぶしぶ	問い	答え	家賃	払
彼	1	0	1	0	0	0	0
しぶしぶ	1	1	0	0	0	0	0
問い	0	1	1	0	0	0	0
答え	0	1	1	1	0	0	0
家賃	0	1	1	0	0	0	0
払	0	1	1	0	0	1	0

Table 1: Resulting Matrix

ten those words follow “彼”, though not necessarily immediately. Notice the absence of particles such as は, “wa”, and を, “wo”, in the matrix. Although these occur frequently, they tend to reduce prediction accuracy of the critical words; then again, they are required in just about every sentence. We’ll discuss how we handle them further in the text.

4 Evaluation

As a test for suitability, we wanted to evaluate how useful this system would be for a particular writer. The basic metric for success was a reduction in the number of keystrokes required to construct an entire article. By this definition, keystrokes also include the “space” and “enter” keys that are used to scroll and make selections among candidate words.

For this exercise, we gathered 10 recent blog entries of a particular writer for a Japanese technology website and analyzed the text to produce a (large) matrix of word frequencies. Then, using this corpus, we went on to simulate this same user writing two new articles. The two articles being simulated are, naturally, not included in the corpus.

First, we passed the articles through a kanji to romaji converter which emits phonetic alphabetic strings that would sound the same as the Japanese characters when pronounced. This system gives a useful baseline metric, but loses all grammatic meaning (i.e. a reader of this alphabetized Japanese can’t easily tell where words begin or end).

Second, we type the articles using Kotoeri (a Japanese input system for the Mac), which gives no prediction. Next, we type them again using POBox, which offers bigram prediction. Finally, we re-type yet again using our system.

4.1 Results

Initial evaluation and analysis of our predictor show some reduction in the number of keystrokes needed to produce the articles. In the first case, the short article did not contain many of the words already used in the corpus, and many words were used only once, so the system didn’t have much chance to reduce keystrokes. There is however, significant reduction in the second case. This time, upon comparison with the corpus, the latest article contained a number of preexisting words; in

System	Short Blog	Latest Blog
romaji	1210	2319
kotoeri	1374	2780
pobox	1293	2433
predictor	1262	2018

Table 2: Keystrokes needed to produce articles.

addition, a good number of words and sentence fragments were repeated, giving the system good opportunities to make useful predictions.

Summarized, for the short article, POBox and our Predictor were able to shave 6% and 12%, respectively, off the total number of keystrokes the author would have needed using Kotoeri alone. For the longer article, the savings are 8% and 27.41%, respectively.

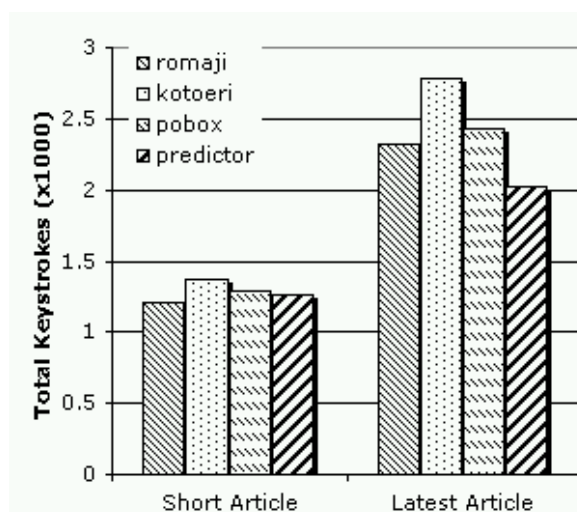


Figure 5: Keystrokes needed to produce articles.

Although an existing corpus of relevant text is useful in improving accuracy, we have noted (per Figure 4 and Table 1) that with two or three related sentences, the number of keystrokes needed to produce new, related sentences can also be significantly reduced.

4.2 Dealing with Particles

With an initial focus on independent words (i.e. verbs, nouns, adjectives), we quickly discovered that the priorities for word particles (e.g. “が, ga” “を, wo”) got pushed down. In one particular example, after pressing the initial ‘g’ for the particle “ga”, we needed to scroll past 11 other choices (13 keypresses in total) before arriving at the hiragana representation. Normally, using the Kotoeri input method (without prediction), this takes all

of 3 strokes. Although these particles occur quite frequently in Japanese text, simply adding them to the prediction matrix would unnecessarily increase keystrokes as the user would often have to scroll past these. The solution involved taking another look at the structure of the language.

In Japanese, a typical short sentence might have the following form (this example is taken from a blog entry in our corpus):

日本 から 手紙 が 来た。
 Nippon kara tegami ga ki ta
 Japan from letter (subject) arrive -d

IW AW
IW AW
IW AW
BB
BB
BB

IW: independent word
 AW: ancillary word
 BB: basic block

From this brief analysis, our system tries to predict where ancillary words might occur and puts them at the top of the prediction list when they are most likely to be used.

5 Future Work

Since there was no automated way of simulating a user typing arbitrary text, the contents of the two articles were manually typed, and we interacted with our system (much as the original author might) to make selection choices in real time to match the words in the articles. This tedious effort illustrates the need for an automaton that can scan a list of predictions and choose the word that best matches its counterpart in the example article.

Currently, there is no way to indicate to our system that a previously chosen word was eventually deleted from an article: perhaps the author has chosen to use a different word in place of a previous prediction. In its current form, even though the word will eventually become lower in priority as other words are used in its place (and thus fall lower in the list of candidate words), our system still keeps it and all relevant structures around.

There are also some memory and CPU performance issues that arose from this test implementation. In some instances, the user had to wait up to 1 second between typing a letter and seeing a list of candidates as the system compiled its predictions. As this is a prototype system, we have

already indicated a number of places where speed might be improved e.g. caching prediction and selection results, limiting the size of the array to the most recent 50 words or so, and performing some sort of paging to bring in only portions of a corpus that has some relevance to the current paragraph fragment.

Another improvement in performance could be achieved by redesigning the system to be more monolithic in structure. Currently the system is a hybrid of existing subsystems, done mainly to reduce development time. However, this approach apparently reduces the overall efficiency, as observed in our evaluation. Particularly, the redundancy of dictionary is the most notable problem; currently, the matrix in our system, the dictionary of POBox and the PoS dictionary of MeCab all contain basically the same word information. If we could combine these three dictionaries into one, both memory usage and search performance could be improved significantly.

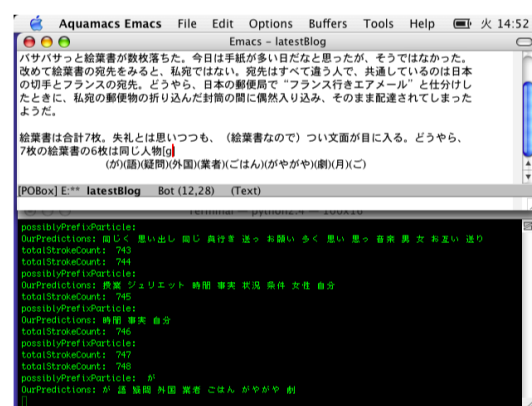


Figure 6: Screenshot

References

A. Copestake. 1997. Augmented and alternative nlp techniques for augmentative and alternative communication. In *Proceedings of the ACL workshop on Natural Language Processing for Communication Aids*, pages 37–42.

Yumi Ichimura, Yoshimi Saito, Kazuhiro Kimura, and Hideki Hirakawa. 2000. Kana-kanji conversion system with input support based on prediction. In *Proceedings of the 18th conference on Computational linguistics*, pages 341–347, Morristown, NJ, USA. Association for Computational Linguistics.

Hiroiyuki Komatsu, Satoru Takabayashi, and Toshiyuki Masui. 2005. Corpus-based predictive text input. In *Proceedings of the Third International Conference on Active Media Technology*, Kagawa, Japan, May. IEEE Press.

Toshiyuki Masui. 1998. An efficient text input method for pen-based computers. In *Conference on Human Factors in Computing Systems, CHI'98*, pages 328–335.

Toshiyuki Masui. 1999. Pobox: An efficient text input method for handheld and ubiquitous computers. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 288–300, London, UK. Springer-Verlag.

R. William Soukoreff and I. Scott MacKenzie. 2003. Input-based language modelling in the design of high performance text input techniques. In *Graphics Interface*, pages 89–96. CIPS, Canadian Human-Computer Communication Society, A K Peters, June. ISBN 1-56881-207-8, ISSN 0713-5424.

Kumiko Tanaka-Ishii, Yusuke Inutsuka, and Masato Takeichi. 2002. Entering text with a four-button device. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Tegic Communications, Inc. 2000. T9.