Corpus-based Predictive Text Input

Hiroyuki Komatsu
Tokyo Institute of Technology
Ookayama 2-12-1 W7 1F 102,
Meguro-ku, Tokyo, JAPAN
komatsu@matsulab.is.titech.ac.jp

Satoru Takabayashi
National Institute of Advanced Industrial Science and Technology
2-28-8 Honkomagome,
Bunkyo-ku, Tokyo, JAPAN
s.takabayashi@aist.go.jp

Toshiyuki Masui
National Institute of Advanced Industrial Science and Technology
2-28-8 Honkomagome,
Bunkyo-ku, Tokyo, JAPAN
t.masui@aist.go.jp

Abstract

We introduce a new predictive text input system that uses visited documents for predicting the user's next input word. With our method, users can efficiently compose new texts using other documents as dictionaries for the input prediction. We have developed a document storage system Kukura that stores all the texts visited by users and cooperates with our predictive input system PRIME. In this paper, we describe the design, implementation, and evaluation of our approach.

1. Introduction

A predictive text input system predicts the user's next input word from the characteristics of natural languages and the user's text input history. It can dramatically reduce the burden of text input tasks [1, 2, 7, 9], especially in environments where standard full-size keyboards cannot be used. When a user of a predictive text input system types the "a" key and "p" key to enter "application", the system suggests "apple", "application", and other words which begins with "ap" so that the user can easily select "application" from the list (Figure 1, 2).

Predictive text input systems are now widely used on Japanese mobile phones and PDAs, with which users can enter Japanese Kanji characters only by specifying a small portion of the pronunciation of the input word. Candidate words are usually selected based on the word frequencies and the user's usage pattern, but it would be better if the system can predict words based on the context of the text composition task. For example, when a user is writing an email message to his friend, names of the people and places they know would be better candidates than place names where they've never visited.

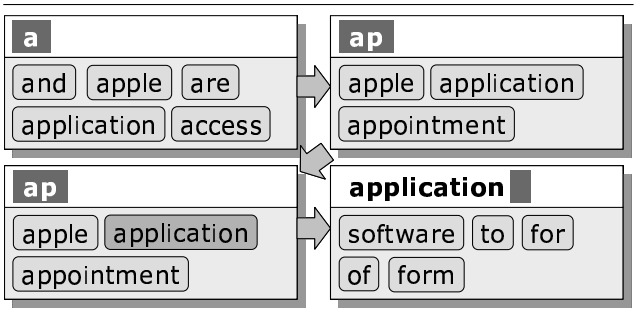

Figure 1. Predictive input in English (typing "application").

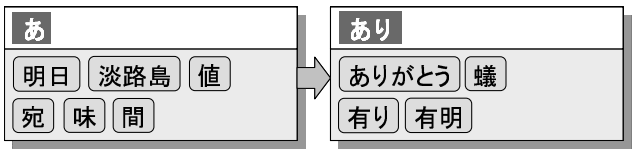

Figure 2. Predictive input in Japanese.

In this case, the system can use old email messages exchanged between them for the prediction. The system can also use the text displayed on the Web browser just before composing the message, since there can be some relation between the contents of the page and the text of the message.

We have been developing a predictive text input system which reflects the context of the user's composition task, using a document storage system that stores all the texts accessed by the user (Figure 3).

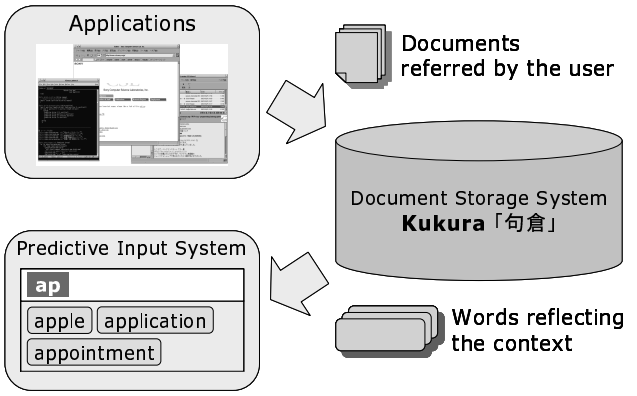

Figure 3. Document storage system for predictive input system.

2. Predictive Input with a Document Storage System

In order to perform context-based prediction, we have been developing a predictive text input system *PRIME* and a document storage system *Kukura*. PRIME is a general-purpose predictive text input system, and it uses the data stored in Kukura for context-based word prediction. Figure 4 shows the flow of data exchanged between PRIME, Kukura, applications and the user.

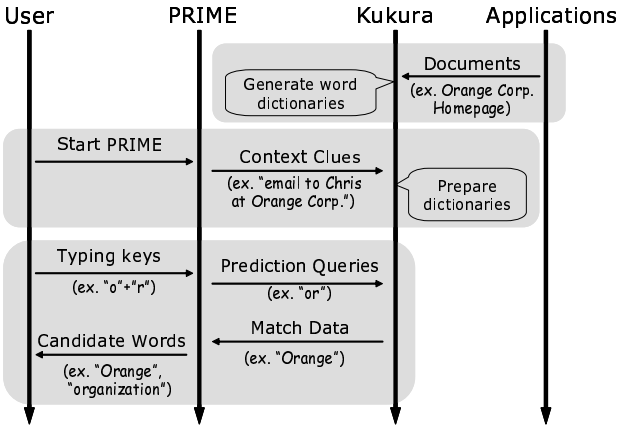

Figure 4. Process flow diagram between components.

2.1. PRIME

We have been developing a general-purpose predictive input system PRIME[4](Figure 5), and distributing it as a free software under the GPL2 license [1]. PRIME runs on Unix, MacOS, and Windows, and can be used for Japanese and English text input.

Figure 5. Using PRIME on X11.

Unlike other predictive text input systems, PRIME can generate more than one words as one candidate. For example, PRIME can predict a combined word like “context-based” from “con” or “context-b”.

2.2. Kukura

Kukura is a document storage system that stores all the text data accessed by users. When a user browses a Web page or reads a text document, words in the document are saved into the Kukura database. Application programs can communicate with Kukura to access the database, and use it to predict the user’s next input (Figure 3). Since Kukura keeps all the word occurrences of visited documents, it can be used as an information retrieval system, just like Google’s Desktop Search system[2].

In our current implementation, Kukura can handle Web pages accessed by the user, exchanged email messages, chat logs of instant messengers, and document files under specified directories.

2.3. Compound Word Generation

For effective word prediction, Kukura generates compound words like “John Smith” and “document storage sys-

1 http://sourceforge.jp/projects/prime/
2 http://desktop.google.com/

tem" in addition to single words, because compound words tend to reflect the context of the document better than single words. For example, while words like "document", "storage" and "system" are common to many documents, "document storage system" reflects the characteristics of a document like this paper.

Kukura generates compound words in English and Japanese, using a POS (part of speech) analyzer and morphological analyzers, in addition to heuristic knowledge of both languages. After Kukura gets a list of words with those POS from those analyzers, it constructs compound words from the list. For example, when Kukura gets the list ["This (pronoun)", "is (verb)", "a (article)", "document (noun)", "storage (noun)", "system (noun)"] from "This is a document storage system", it generates a compound word "document storage system" from the grammatical information.

2.4. Flexible Priorities of Words

Kukura attempts to store all the documents and extract all the words from them so that PRIME could predict appropriate words. However, the priorities of the candidate words generated by Kukura are kept lower than general words registered in the standard prediction dictionary. This is a preferable strategy because users usually do not reuse all the text they have seen.

For example, when a user is composing an email message about going to a restaurant, she certainly wants to type the name of a restaurant she has seen on a web page, but she would not want to type it in other situations. Because of this reason, PRIME changes the priorities of words generated by Kukura according to the user's context.

3. Implementations

In this section, we show the implementation details of Kukura and PRIME.

3.1. Collecting Data

Kukura takes three different strategies for collecting text data from the documents accessed by the user.

- Checking Special Directories

  Users can tell Kukura to check special directories which holds cache or log files used by applications. Web browsers and instant messengers usually keeps those files in special directories, and Kukura can check those directories periodically and find new and updated files to update the dictionary database.

- Using Proxy Systems

  Proxy systems are also used for extracting texts from applications. In our current implementation, an HTTP proxy and a command line proxy which logs all the shell commands are used by Kukura. Using proxy systems is sometimes better than checking special directories, since the system can collect data as soon as they are used.

- Modifying Applications

  We can also modify the behavior of each application to communicate with Kukura directly. We are using an Elisp (Emacs Lisp) program which directly communicates with Kukura, so that Emacs can use all the features of Kukura.

3.2. Handling Compound Words

Since compound words reflect the context of a document better than single words, Kukura automatically generates compound words in both English and Japanese.

Kukura basically consults the POS value of each single word to generate compound words in common, and executes POS-and-morphological-analyzers to detect those POSes. Kukura uses Wordnet[8] to detect POSes of words in English, and uses MeCab[6] for Japanese.

In the current implementation, after calculating the POS of each word, Kukura combines both noun, prefix and suffix words and a part of unregistered words like "AMT2005" and "セカチュー" into compound words. Using this method, Kukura produces context-sensitive words such as "Kagawa University" and "電車男".

3.3. Predictive Input Using Stored Documents

PRIME can use multiple prediction engines at the same time, and it can select one of the engines to communicate with Kukura (Figure 6).

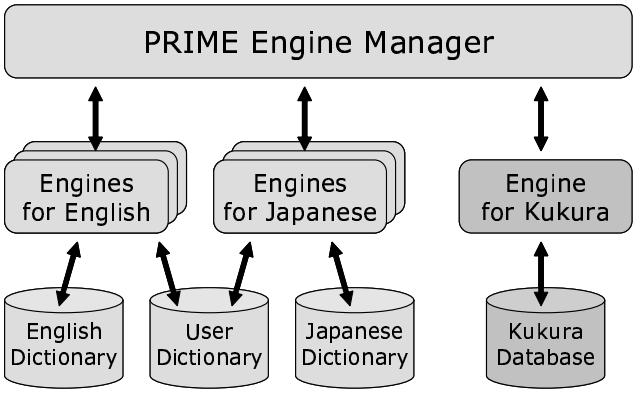

Figure 6. Multiple engines management by PRIME.

A prediction engine for Kukura collects the context information from the user's input history, the target application, and its properties. For example, if a user is exchanging messages with her friend, the prediction engine collects her input history, the name of that chat software, and the user ID of her friend.

4. Evaluations

We evaluated the effectiveness of our strategy with three data sets:

- chat logs recorded over six months,
- results of compound words generated by Kukura, and
- input logs recorded by PRIME, usred with Kukura.

In this section, we first describe the reuse rate of words in chat logs, and then we show examples of the generated compound words. We will discuss the actual usage of PRIME and Kukura at the end.

4.1. Reuse Rate of Words

To prove that predicted words generated by Kukura actually reflect the user's input context, we calculated the word reuse rate from a chat log exchanged between two users. We used a chat log of MSN Messenger[3] and calculated the reuse rate with the following steps (Figure 7):

1. Remove controls tags from the log
2. Split the log by 1.5KB, and create a set of text files $F = [f_1, f_2, ..., f_n]$.
3. Generate a set of dictionaries $D = [d_1, d_2, ..., d_n]$ from F; each dictionary d_i is generated from f_i.
4. Generate another set of dictionaries $E = [e_1, e_2, ..., e_n]$; each dictionary e_i is generated from the combination of $[d_1, d_2, ..., d_{i-1}]$.
5. Get the common words c_i among d_i and e_i
6. Calculate the word reuse rates by diviging the number of word in d_i ($equal\ sd_i$) by the number of words in c_i ($equal\ sc_i$).

We measured the reuse ratio using two chat logs recorded over six months. Figure 8 shows the reuse rates calculated from those chat logs.

Both of the graphs show that reuse ratio can reach as high as 50% over time. Some of the most reused word are "XYZ 空間 (XYZ space)", "日本語スペルチェッカ (Japanese spelling checker)", and "C プログラミング (C programming)".

The reuse ratio occasionally goes down to 20%, when 1) the topic of the conversation changed completely, or 2)

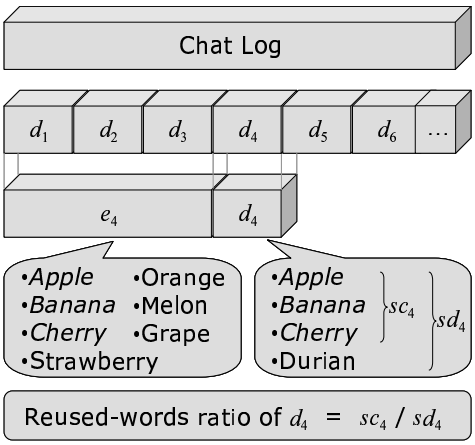

Figure 7. How to evaluate ratios of reused words on chat logs.

Kukura failed to parse the text which consists of colloquial expressions and text-oriented figures like Figure 9.

4.2. Compound Word Examples

We could find the following compound words in the Kukura database:

- Software Engineer
- Unix programming
- Mountain View (city in California)
- お気に入り-カフェ ("favorite cafe")
- 簡易-乾燥-機-付き ("with tiny dryer")
- 山-廃-純-米-吟醸 (Japanese sake of special make)

Those words actually reflect the context of documents.

However in our current implementation, Kukura understands neither preposition (e.g. "of") nor postpositional particle words (e.g. "の") in Japanese, so Kukura fails to generate compound words like "Amulet of Yendor" and "徹子の部屋". We are developing the new version of Kukura which can handle these types of compound words.

4.3. Usage Experiences

We collected a log of user input with PRIME and Kukura in Japanese [4], and Table1 shows some of the words input by the user. All of those words were not found in the original PRIME dictionary, but generated by Kukura after the user's text input operations.

"アダージョ" is the name of a local restaurant, and "料理写真日記" is the title of a blog. They are too special to

3 http://messenger.msn.com/

4 Correctly, the user used OpenPOBox which is the former version of PRIME.

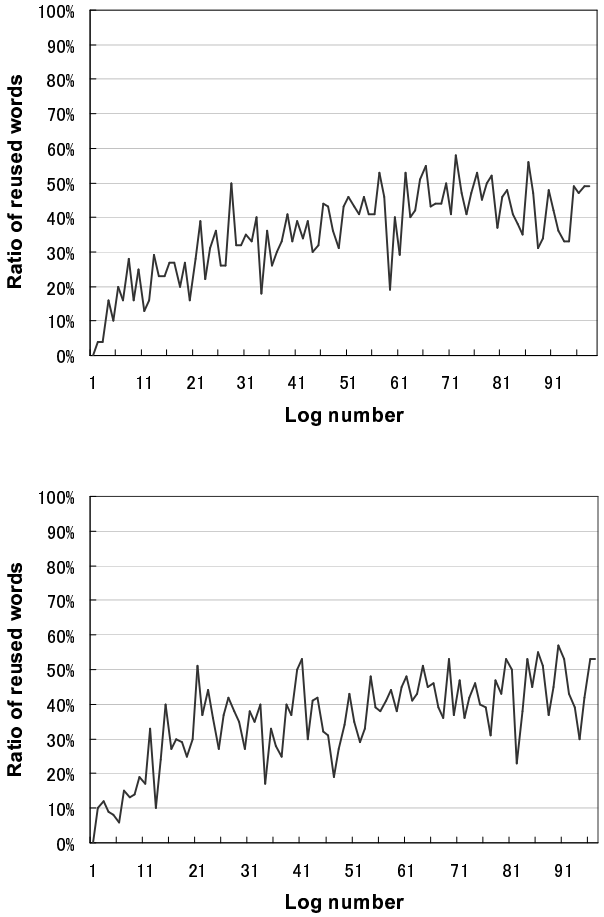

Figure 8. Ratios of reused words on chat logs.

be registered in the standard word dictionary of PRIME, despite the fact that it can contain more than 250,000 words. [5] Most of the users never have a chance to use these words, but Kukura could automatically register them to the dictionary and make them available to the user.

The rest of the words in the table are also compound words, and all of the words are predicted from the user input before the user had actually typed the complete pronunciations. If the user had not used Kukura, he would have needed to type those whole pronunciations. These facts show that Kukura successfully generates compound words from the context.

5 the PRIME dictionary is the biggest dictionary in free software Japanese input systems now.

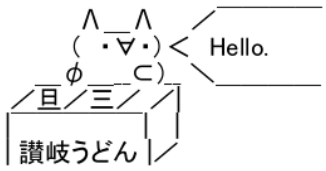

Figure 9. A sample of text-oriented figures.

Inputted word	Pronunciation (**Typed characters**)
アダージョ	**ada**-*jo*
アルバイトさん	**aru***baitosann*
同音語辞書	**douo***onnigigojisho*
校正ツール	**kou***seitu-ru*
プロモーションビデオ	**pu***romo-shonnbideo*
料理写真日記	**ryo***urishasinnnikki*
かな漢字変換ソフトウェア	**kana***kannjihennkannsofutowea*

Table 1. Automatically created input words.

5. Related Work

Nanashiki[5] is another predictive text input system we have developed. Nanashiki focuses only on the current document the user is composing, because the current document is supposed to have the context the user is typing (Figure 10).

Nanashiki dynamically extracts candidate words from the current document and use them for prediction, so that it can make good use of changing texts such as instant messenger's logs.

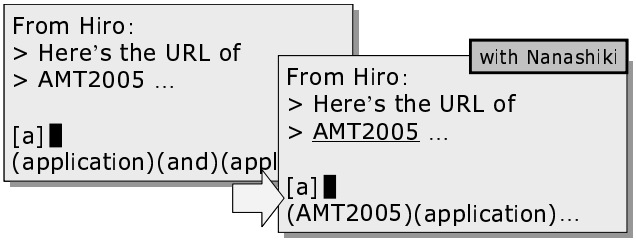

Figure 10. Word prediction by Nanashiki.

Japanist[1] is another predictive input system by Fujitsu which also generates a word dictionary from the user's documents. Kukura generates its dictionaries automatically, while Japanist requires the user to specify which documents should be used for the prediction.

Google Desktop Search[3] provides a local document search and merges it into the web search by Google, so that a user can search document files read by the user with

the same interface as web search. If Google Desktop Search would generate word dictionaries from its crawled files, predictive input systems could utilize them instead of using Kukura.

6. Conclusions

We proposed a new predictive text input system integrated with a document database which collects all the text browsed by the user. This system predicts and suggests more context-sensitive words to the users, and generates word dictionaries from the browsed documents.

Since the next version of Microsoft Windows (LongHorn) attempts to provide a local file search system in a user's PC like Google Desktop Search, almost all computers will be able to perform document search much more easily than using current standard personal computers. We believe that this kind of innovative text retrieval technologies can be of great help to efficient text input in the future.

References

[1] Fujitsu Limited. Japanist, 2000. http://software.fujitsu.com/jp/japanist/.

[2] T. Fukushima and H. Yamada. A predictive pen-based japanese text input method and its evaluation. *IPSJ Journal*, 37(1):23–30, 1996.

[3] I. Google. Google desktop search, 2004. http://desktop.google.com/.

[4] H. Komatsu. Prime: Predictive input method editor, 2002. http://taiyaki.org/prime/.

[5] H. Komatsu, S. Takabayashi, and T. Masui. Context-aware predictive text input method using dynamic abbreviation expansion. *IPSJ Journal*, 44(11):2538–2546, 2003.

[6] T. Kudo. Mecab: Yet another part-of-speech and morphological analyzer, 2001. http://chasen.org/˜taku/software/mecab/.

[7] T. Masui. An efficient text input method for pen-based computers. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '98)*, pages 328–335. Addison-Wesley, April 1998.

[8] G. A. Miller. Wordnet. http://www.cogsci.princeton.edu/˜wn/.

[9] K. Tanaka-Ishii, Y. Inutsuka, and M. Takeichi. Entering text using a four button device. In *19th International Conference on Computational Linguistics*, pages 988–994, 2002.