Migemo: Incremental Search Method for Languages with Many Character Faces

Satoru Takabayashi[†‡] and **Hiroyuki Komatsu**[*] and **Toshiyuki Masui**[†]
[†]Sony Computer Science Laboratories, Inc.
[‡]Nara Institute of Science and Technology, Graduate School of Information Science
[*]Tokyo Institute of Technology, Graduate School of Information Science and Engineering

satoru@csl.sony.co.jp, komatsu@matsulab.is.titech.ac.jp, masui@csl.sony.co.jp

Abstract

We introduce a new incremental search method called *Migemo* for languages with many character faces. Migemo performs the incremental search by dynamically expanding the input pattern into a compact regular expression which represents all the possible words that match the input pattern. We show that Migemo is useful not only for searching texts in Japanese and other East Asian languages, but also for performing sophisticated searches on ASCII-only documents.

1 Introduction

Incremental search is one of the most powerful operations provided in text editors like Emacs, as the simplest form of dynamic query. Various text matching algorithms can be used for implementing incremental searches for languages with ASCII characters, but they are not directly applicable to Japanese and other East Asian languages, where keyboard characters do not directly correspond to text characters.

In conventional Japanese text editors, users have to select Japanese characters before performing a search. In this case, the advantage of incremental search is almost lost, because Japanese character entry usually takes multiple steps like the following:

1. Type the pronunciation of a word using an ASCII keyboard.
2. Convert the ASCII text into a Kana text which represents the pronunciation of a Japanese word.
3. Convert the Kana text into a set of Kanji words by a Kana-Kanji converter. One pronunciation usually corresponds to more than one Kanji characters. For example, "機械 (machine)", "機会 (opportunity)", and "奇怪 (strange)" all have the same pronunciation "kikai".
4. Select the desired Kanji word from the set of candidate words.

If users want to find "奇怪" with a conventional incremental search method, they have to type more than ten keys[1]. Figure 1 shows the process of selecting "奇怪" as the search keyword. A search with Kana-Kanji conversion in this way is thus not dynamic at all.

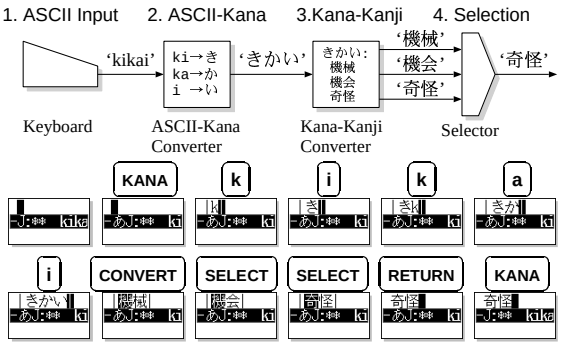

Figure 1: The process of selecting "奇怪 (strange)" as the search keyword.

2 Migemo

We propose a new incremental search method called Migemo, which solves the problem described in the previous section. Figure 2 shows the process of performing incremental search for a Japanese word "奇怪" with the Migemo for Emacs[2].

In this example, the user typed only four keys (^S k i k) to find "奇怪", which is much easier than typing twelve keys in the previous example. Unlike conversion-based search, the search with Migemo is truly incremental and dynamic, just like an incremental search for ASCII documents is.

[1]Using Emacs, the users have to type ^S KANA k i k a i CONVERT SELECT SELECT RETURN KANA: ^S starts the incremental search, KANA starts and ends Kana-Kanji conversion, CONVERT converts a Kana text to a set of Kanji words, SELECT selects a Kanji word from the candidates of the Kanji words, and RETURN finishes the selection process of Kanji words.

[2]http://migemo.namazu.org/.

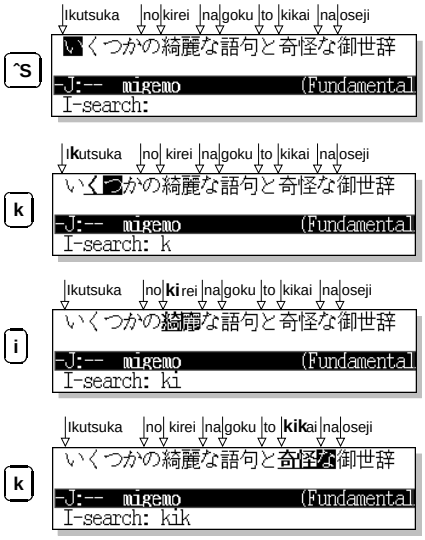

Figure 2: The process of an incremental search for a Japanese word “奇怪 (strange)” with Migemo.

2.1 The Method

Migemo skips the Kana-Kanji conversion by dynamically expanding the input pattern into a compact regular expression which represents all the possible words that match the input pattern. Figure 3 shows the expanded regular expressions for “k”, “ki”, and “kik”.

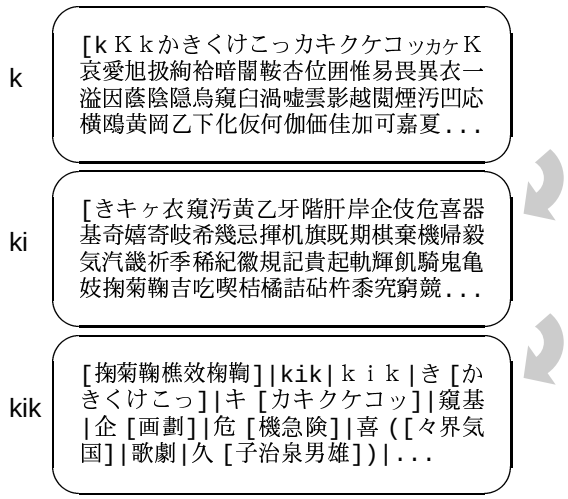

Figure 3: Expanded regular expressions for “k”, “ki”, “kik”.

Although the actual search is performed with the expanded regular expressions, the user does not have to care about the expansion process because the expansion is performed silently in the background.

2.2 Regular Expression Expansion

Migemo generates a regular expression(Hopcroft and Ullman, 1979) using a dictionary and expansion rules. The first regular expression in Figure 3 includes the candidates generated by ASCII-Kana conversion rules to cover all the Japanese Hiragana and Katakana characters beginning with “k”:

k ⇒ か | き | く | け | こ |っ| カ | キ | ク | ケ | コ |ッ|ヵ|ヶ

Extraction of the Kanji words is performed by looking up the words in an ASCII-Kanji dictionary. Figure 4 shows an excerpt from our dictionary.

Pronunciation	Kanji Representation
kikai	機械 機会 奇怪 器械 貴会 気塊 喜界
kikaiabura	機械油
kikaibuhin	機械部品
kikaichinou	機械知能
kikaigakkai	機械学会
kikaigaku	機械学

Figure 4: Dictionary for the regular expression expansion (excerpt).

All the Kanji words which match the given pattern are extracted from the dictionary and connected together by the *or* operator “|” to generate the regular expression. The search performance depends on the efficiency of the regular expression matching, and Migemo optimizes the *or*-connected regular expressions by unifying redundant portions of the words. The optimization allows NFA (nondeterministic finite automaton) regular expression engines to eliminate redundant comparisons(Friedl, 1997). Figure 5 shows the optimization for an input pattern “nez”.

Naive Regular Expression

nez|ｎｅｚ|寝|寝|根魚|根崎|寝醒め|根差|根差し|寝惚|寝相|根津|禰津|鼠|鈬|鼠色|鼠男|鼠達|鼠取|捻|捩|螺|捻子|螺子|捩子|ネジ|捻じ伏|拗|捩じ込み|根占|捩じり鉢巻き|捩り鉢巻き|根城|ねざ|ねじ|ねず|ねぜ|ねぞ|ねっ|ネザ|ネズ|ネゼ|ネゾ|ネッ

Optimized Regular Expression

[寝鼠捻螺寝拗捩鈬]|nez|ね[ざじずぜぞっ]|ネ[ザジズゼゾッ]|根[魚差崎城占津]|禰津|ｎｅｚ

Figure 5: Optimization of the regular expression for “nez”.

In the example, words beginning with “鼠” such as “鼠 (mouse)”, “鼠色 (gray)”, and “鼠男 (rat man)” are

unified into the shortest word “鼠”, and words consisting of a single character such as “寝 (sleep)”, “螺 (screw)”, and “捻 (twist)” are grouped into the character class “[寝鼠捻螺寝拗捩角]”. Moreover, words having a common prefix such as “根 魚 (fish)”, “根 差 (take root)”, “根 崎 (name of a town)” are gathered into “根[魚差崎城占津]”.

More than one ASCII strings sometimes represent the same Kana string. For example, both “kanji” (so-called Hepburn fashion) and “kanzi” (so-called Kunrei fashion) represent “かんじ”. Migemo allows both representations to generate the regular expressions.

2.3 Pre-Compiled Regular Expressions

Tens of thousands of Japanese words have pronunciations which begin with “k”, “s”, and “t”[3]. Since it takes a long time to compile regular expressions for all the words with these consonants, pre-compiled regular expressions are prepared for these short patterns.

3 Applications

3.1 Personal Information Management

Incremental search is particularly useful for personal information management (PIM) systems. We implemented a PIM system called *Q-Pocket*(Masui, 2000)[4] for PalmPilot, which uses Migemo for full-text retrieval. It is much easier to search Japanese texts from direct Graffiti input than selecting a Japanese keyword before performing the search. Here, a Japanese dictionary used in the mobile text input system *POBox*(Masui, 1998) is used for Migemo. Using the same dictionary both for text input and for text search has an advantage of consistent text handling. If a word “★” is defined to have a pronunciation “star” in the dictionary, a user can type [s][t][a][r] to enter “★” using an indirect text input method shown before. Documents including “★” can be retrieved by typing [s][t][a][r]. To realize fast full-text searching, Q-Pocket employs Aho-Corasick(Aho and Corasick, 1975) method for handling *or*-connected regular expressions. Figure 6 shows how incremental full-text search of memos and schedule data is performed with Q-Pocket.

3.2 Language-Independent Incremental Search

Applications of Migemo are not limited to handling documents in Asian languages, but it is useful for handling ASCII documents. For example, if users can use an abbreviation dictionary that includes entries like:

jfk John F. Kennedy
jfk John Fitzgerald Kennedy

[3]In our dictionary, 47,292 words have pronunciations beginning with “k”, out of 153,517 words.

[4]http://www.csl.sony.co.jp/person/masui/QPocket/Palm/.

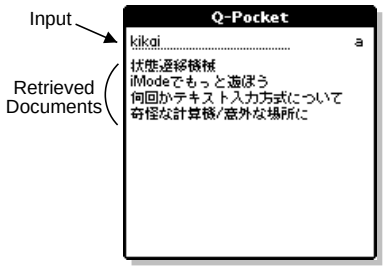

Figure 6: Performing incremental full-text search of memos and schedules with Q-Pocket.

they can type [j][f][k] to find “John F. Kennedy” and “John Fitzgerald Kennedy”. If users of an HTML editor can use a dictionary which includes entries like:

header <h1>
header <h2>
header <h3>

they can incrementally search for all the header tags by typing [h][e][a][d][e][r].

Figure 7 shows how to search ‘你好’ (“hello” in Chinese) by typing [h][e][l][l][o] using an English-Chinese dictionary including an entry:

hello 你好

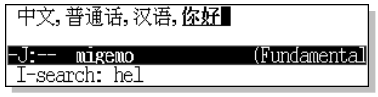

Figure 7: Cross-lingual incremental search using an English-Chinese dictionary.

3.3 Dynamic Abbreviation

Dynamic abbreviation is a method to re-input previously typed words with lower typing costs[5]. For example, one can re-input “floccinaucinihilipilification” by typing only [f][l][o][c][EXPAND]. The [EXPAND] key expands “floc” into “floccinaucinihilipilification”. Dynamic abbreviation is not only useful for reducing the typing costs, but also useful for reducing spelling errors.

However, conventional dynamic abbreviation does not work effectively for Japanese texts because of the necessity of Kana-Kanji conversion. To realize Japanese dynamic abbreviation, we applied the technique of the regular expression expansion provided by Migemo. Since dynamic abbreviation is performed by searching the text backward for candidate words from the cursor position, we use the regular expression generated by Migemo for the search to skip Kana-Kanji

[5]In Emacs, dynamic abbreviation is called as “dabbrev”.

conversion. Figure 8 shows how the dynamic abbreviation with Migemo is performed.

Figure 8: Dynamic abbreviation: expansion of “do”.

4 Discussion

4.1 Handling of Homonyms

The advantage of Migemo is that it skips Kana-Kanji conversion in incremental searches. On the one hand, Migemo can tolerate inconsistency of word representation. For example, a user can find “こんにちは (hello)” and “今日は (hello)” by the same pattern “konnnichiha”. On the other hand, Migemo might find inappropriate words accidentally because of the homonyms. For example, a user might find “機械 (machine)” instead of “奇怪 (strange)” when the user want to find “奇怪” because both “機械” and “奇怪” have the same pronunciation “kikai”. This is a trade-off of our method. In our experience of using Migemo for several months, the latter situations rarely occur. Even though the inappropriate matching occurs, the user can simply ignore it by proceeding to the next matching position. As a last resort, the user can get back to the conventional incremental search method to avoid the ambiguity of homonyms.

4.2 Use of a Dictionary

Since Migemo uses a dictionary to generate regular expressions, Migemo cannot find words that are not defined in the dictionary, just like a Kana-Kanji conversion system cannot convert a Kana string to a word which is not defined in the dictionary.

Since the latest version of Migemo only uses words in the dictionary, it cannot find a phrase which consists of more than one words. For example, Migemo cannot find a phrase like “現在の実装 (current implementation)”. We are planning to extend the algorithm so that it can handle phrases. In Kana-Kanji conversion, handling of phrases is a hard task because the conversion has to disambiguate the phrases as natural as possible. However, handling of phrases for Migemo can easily be realized by simply gathering all possible candidates and connecting them by the *or* operator to generate regular expressions.

4.3 Regular Expression Expansion

While Migemo performs Japanese incremental search by expanding the input pattern into regular expressions, similar search is also possible if Japanese texts are converted to ASCII texts before the search.

The simplest method is to convert the whole Japanese text into the internal ASCII text at every search. Incremental search can be realized by searching the internal ASCII text. For example, a Japanese text “テキストと漢字” is converted into “tekisuto to kanji”. However, since the conversion requires time linear in the length of the text, converting the large text at every search is thus not practical.

Another method is to convert the partial text dynamically into the internal ASCII text as proceeding incremental searches. However, the method requires a rather complicated algorithm to implement. Moreover, the Kanji-ASCII conversion is not always accurate because of ambiguity of Kanji expressions.

5 Conclusions

We introduced a new incremental search method called Migemo for handling languages with many character faces. Migemo can be used in various situations including incremental searches in text editors and text retrieval in PIM systems. We are planning to apply the same technique for wider range of search tasks, including incremental approximate string matching, and image retrieval from keywords.

References

Alfred V. Aho and Margaret J. Corasick. 1975. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340.

Jeffrey E. F. Friedl. 1997. *Mastering Regular Expressions*. O’Reilly.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Toshiyuki Masui. 1998. 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.

Toshiyuki Masui. 2000. Q-Pocket: A new approach to personal information management. In *Interactive Systems and Software VIII: Japan Society for Software Science and Technology WISS2000*, pages 191–196, December.