

動的略語展開を利用した文脈をとらえた予測入力

小松 弘幸[†] 高林 哲^{††,†††} 増井 俊之^{††}

予測入力システムとは、自然言語の知識や操作履歴などから利用者の入力単語を予測することにより、少ない手間での文章入力を支援する入力システムである。近年、予測入力システムは携帯電話や携帯情報端末などで広く利用されている。このような予測入力システムでは、一般的な自然言語の統計情報や利用者の操作履歴を利用して予測が行われるため、利用者が今まで使用したことがなくかつ一般的でない単語は予測候補となりにくいという問題があった。たとえば、メールやインスタントメッセージを利用して情報を交換する場合は、話題に応じて特殊な単語がやりとりされることが多いため、この問題が顕著となる。本論文ではこの問題を解決するために、メールの相手のメッセージに含まれる単語など、他の文脈情報も予測に反映させることによって、文脈をとらえた予測を行う手法を提案する。

Context-aware Predictive Text Input Method using Dynamic Abbreviation Expansion

HIROYUKI KOMATSU,[†] SATORU TAKABAYASHI^{††,†††}
and TOSHIYUKI MASUI^{††}

Predictive text input systems reduce the cost of input by predicting user's input using the knowledge of natural languages and the history of user's operations. In recent years, the predictive text input systems have become popular especially for mobile phones and mobile information appliances. Since this kind of predictive input systems usually use the statistics of natural languages and the history of user's operations, there is a problem that words unused and uncommon are not likely to be chosen for the candidate words. For example, the problem becomes clear if users are doing communication using email or instant messengers because they are more likely to use unused or uncommon words according to their topics. In this paper, we propose a new predictive input method to realize a better prediction by solving the problem using context information of the peer's message.

1. はじめに

自然言語の知識や利用者の操作履歴などを用いて利用者の入力単語を予測することにより少ない手間での文章入力を可能にする予測入力システム^{1)~6)}が近年広く利用されている。このようなシステムは、利用者が入力した単語の部分的な読みなどから入力単語を予測し、複数の候補を利用者に提示して選択させることにより操作手順の軽減をはかっている。たとえば、携帯

電話などに搭載されている予測入力システム POBox²⁾では、利用者が「こん」という読みを入力したとき、「こんにちは」、「コンピュータ」、「今週」などの候補を予測して利用者に提示して選択させる(図1)。また、子音変換システム TouchMeKey⁴⁾では、「hnkn」という子音から「変換」や「半券」などの候補を予測するとともに、POBoxと同様の候補予測も行っている。これらのシステムでは、単語の予測には、単語の一般的な出現頻度の情報やアプリケーションの制約、利用者の操作履歴などが予測に使用される。

すべての文章を自分で書く場合はこの方式が有効にはたらくが、メールに返事をする場合のように、特定の分野に話題が集中しているような場合は、その文脈をとらえた単語の予測も行うことができると都合がよい。たとえば、コンサートへの誘いに返事するメールを書こうとする場合には、利用者が「こん」という読みを入力したとき「コンサート」のような単語が候補

[†] 東京工業大学情報理工学研究科

Graduate School of Information Science, Tokyo Institute of Technology

^{††} 産業技術総合研究所

National Institute of Advanced Industrial Science and Technology (AIST)

^{†††} 奈良先端科学技術大学院大学情報科学研究科

Graduate School of Information Science, Nara Institute of Science and Technology

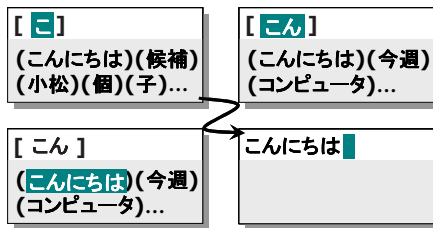


図 1 予測入力例（「こんにちは」の入力）

Fig. 1 Example of predictive text input (input of “こんにちは”).

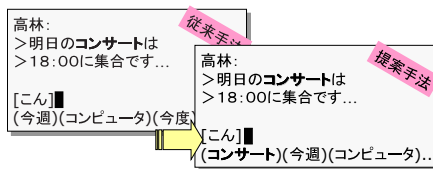


図 2 文脈をとらえた予測入力

Fig. 2 Predictive text input considering the context.

となる方が都合がよいであろう。このように、前述した従来の予測手法に加えて、入力中の文書の背景にある文脈の情報が予測に反映されると便利である。

作成中の文章にすでに含まれている単語を再入力するような場合、再入力の手間を省くために動的略語展開 (Dynamic Abbreviation Expansion)⁷⁾ という手法が利用されることがある。たとえば、テキストエディタ Emacs では、very-long-variable-name のような長い変数名が使われているプログラムにおいて、何度もこの名前を入力するかわりに very のような先頭の数字文字を入力した後で **EXPAND** キー を押すことにより、自動的に very-long-variable-name に展開される。

このような動的略語展開手法を従来の予測入力手法と組み合わせて使うことにより、コンサートに関する返事の場合は「コンサート」が候補単語となりやすく、それ以外の場合は「コンピュータ」が候補単語となりやすいといった、文脈をとらえた予測入力を行うことができるようになる。

しかし、日本語における動的略語展開は、英文にはない日本語特有の問題からこれまで実用的ではなかった。そこで我々は、日本語の動的略語展開手法 Nanashiki 「七色」を開発し、文脈をとらえた予測入力を、Nanashiki を用いて実現した。

2. 動的略語展開の日本語への適用

日本語の動的略語展開を実現しようとする場合、通常の英数字の入力における場合にはない、以下の問題が存在する。

- かな漢字変換
- 単語の切り出し

第 1 の問題は日本語の入力には、英数字の入力とは異なり「読みの入力 変換 確定」というかな漢字変換の手順が必要なために発生する。スムーズな日本語の動的略語展開を実現するには、この手順を省略する必要がある。たとえば「東京工業大学」という文字列を従来の動的略語展開によって入力する場合、「東京」と入力してから「東京工業大学」に展開するといった手順を踏む。しかし「東京」と入力するためには、まずその読みである「toukyou」を入力した後、読みを漢字へ変換しなければならない。すなわち、従来の動的略語展開は、

- (1) 「toukyou」と読みを入力、
- (2) 「東京」に変換、確定、
- (3) 「東京工業大学」へと動的略語展開、

という 3 段階の手順を必要とする。しかし、この動的略語展開を用いた手順を、以下に示す通常の入力手順である、

- (1) 「toukyoukougyoudaigaku」と読みを入力、
 - (2) 「東京工業大学」に変換、
- と比較しても、かな漢字変換が煩雑なため、動的略語展開を用いた手順の入力効率率はほとんど向上しない。

そのため入力の効率化を目指すには、かな漢字変換を省いた、

- (1) 「touky」と先頭の数字文字を入力、
- (2) 「東京工業大学」に展開、

という動的略語展開が望ましい。しかし、そのためには「touky」という文字列から「東京工業大学」という文字列を検索する仕組みが必要である。

加えて、日本語の動的略語展開における第 2 の問題として「単語の切り出し」がある。文章中から単語を切り出す処理は、英語の場合、空白という明確な単語間の区切りがあるため容易である。しかし、日本語の文章には明確な単語間の区切りがないため、文章を適切な単語群に切り分ける方法が必要がある。たとえば、「最寄りの駅は自由が丘です」という文から「最寄り」や「自由が丘」という単語を取得する必要がある。

3. Nanashiki 「七色」

前章で述べた問題を解決するために、我々は日本語

EXPAND キーは動的略語展開を実行するキーを表す。Emacs では **Alt**+**/** に設定されている。

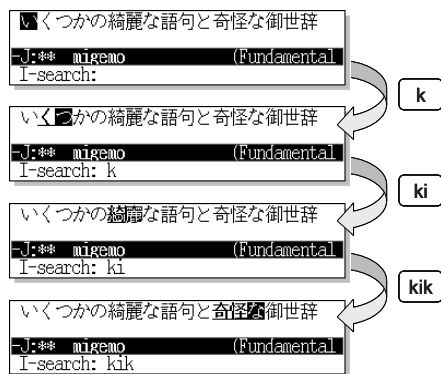


図3 「奇怪」をMigemoでインクリメンタル検索
Fig.3 Incremental search for “奇怪” by Migemo.

の動的略語展開手法 Nanashiki を考案し、実現した。Nanashiki を用いることにより、たとえば、文「最寄りの駅は自由が丘です」に対して入力「jiy」から単語「自由が丘」を取得できる。本章では Nanashiki が行っている、日本語の動的略語展開の実現方法を述べる。

日本語動的略語展開における問題点は、前述のとおり「1. かな漢字変換」と「2. 単語の切り出し」の2点である。我々が開発した Nanashiki はこの2点の問題を解決し実用的な日本語の動的略語展開を実現した。Nanashiki では「1. かな漢字変換」に対しては日本語インクリメンタル検索手法 Migemo を応用して解決した。また「2. 単語の切り出し」に対しては形態素解析システムを用いて解決した。

3.1 かな漢字変換の問題の解決

第1の問題点である「かな漢字変換」の解決、すなわち「kanji」から「漢字」を検索可能にするために、我々が開発した日本語インクリメンタル検索手法 Migemo⁸⁾ を応用した。Migemo は日本語の検索において、かな漢字変換を省略し、ローマ字入力からそのまま日本語のインクリメンタル検索を実現している。図3に Migemo を用いてキーワード「奇怪」をインクリメンタル検索する過程を示す。

この例では、利用者は と入力し、かな漢字変換の作業を省いて「奇怪」の位置に到達している。一方、かな漢字変換をとまなう検索では「kikai」とすべて入力した後に同音異義語の中から「奇怪」を選ぶという手順を踏む。このように Migemo では、かな漢字変換の作業に煩わされることなく、スムーズな日本語のインクリメンタル検索が可能である。

Migemo は利用者が1文字入力するごとに内部で動的に正規表現を展開し、その正規表現を用いて検索を行う。上の「奇怪」に対するインクリメンタル検索のための正規表現は図4のように展開される。最初

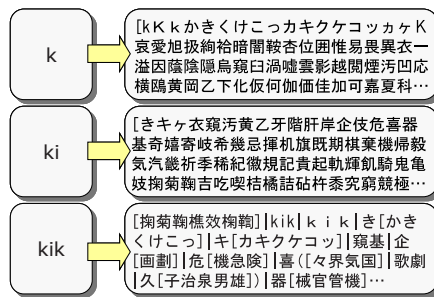


図4 k, ki, kik に対する正規表現
Fig.4 Regular expressions for “k”, “ki” and “kik”.

表1 茶筌の使用例:「最寄りの駅は自由が丘です」の解析結果
Table 1 Example result of ChaSen: analyzing 「最寄りの駅は自由が丘です」.

単語	読み	品詞情報
最寄り	モヨリ	名詞-一般
の	ノ	助詞-連体化
駅	エキ	名詞-一般
は	ハ	助詞-係助詞
自由が丘	ジユウガオカ	名詞-固有名詞-地域-一般
です	デス	助動詞 特殊-デス 基本形

の正規表現では“k”にマッチするテキストの位置へ、中央の正規表現では“ki”にマッチする位置へ、最後の正規表現では“kik”にマッチする位置へと、インクリメンタル検索が進む。

3.2 単語の切り出しの問題の解決

第2の問題点である「単語の切り出し」の問題の解決、すなわち「最寄りの駅は自由が丘です」から「最寄り」や「自由が丘」を取得するためには、形態素解析システムを利用する。形態素解析システムとは、単語辞書や各品詞の接続確率などをもとに、文の構造を解析するものである。表1に、形態素解析システム「茶筌」⁹⁾の使用例を示す。

Nanashiki は形態素解析システムが持つ機能のうち、文を単語ごとに分割する「わかち書き」機能を利用する。

3.3 Nanashiki による動的略語展開の手順

Nanashiki は以下の手順によって、日本語の動的略語展開を行う。

- (1) Migemo による文字列検索。
- (2) 検索された文字列周辺を、形態素解析システムにより単語へ分割。
- (3) 検索された文字列が単語の先頭であれば、動的略語展開の候補とする。

図5は、「今日の東京都の気温」という文章に対して「k」と入力した場合の Nanashiki の動作例である。まず最初に Migemo により、「k」で始まる文字である

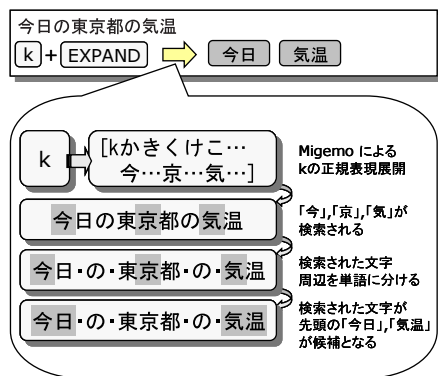


図 5 Nanashiki の処理の流れ
Fig. 5 Process flow of Nanashiki.

「今」、「京」、「気」が検索される．次に形態素解析システムにより，文章を単語に分割し「今」、「京」、「気」がそれぞれ「今日」、「東京都」、「気温」に含まれる文字である情報を得る．最後に，検索した文字列が単語の先頭に含まれることから「今日」と「気温」が動的略語展開の候補であることが分かる．このような手順により，Nanashiki は入力「k」から動的略語展開候補の「今日」と「気温」を取得する．

3.4 Nanashiki の応用例

本論文では Nanashiki を文脈をとらえた予測入力の実現のために利用するが，Nanashiki は予測入力以外にもいくつかの応用例が考えられる．

純粋な動的略語展開

まず，純粋に日本語に対応した動的略語展開機能として使用できる．図 6 は Nanashiki による日本語動的略語展開の例である．利用者の入力 `d` `o` `EXPAND` に対して，まず「動的補完入力」が展開される．繰り返し `EXPAND` を入力すると今度は「Document」が展開される．このように，Nanashiki を用いれば日本語と英語の区別なく動的略語展開が行える．

日本語ファイル名の補完

また，コマンドシェルでの日本語ファイル名の補完機能においても Nanashiki は有効である．ファイル名の補完機能は，動的略語展開機能に似た機能である．たとえば `/tmp/nanashiki.html` というファイルがあり，このファイル名を入力する場面では，利用者は `/tmp/na` と入力した後に `EXPAND` キーを押せば `/tmp/nanashiki.html` と展開されるため，入力の効率化がはかれる．

Nanashiki を用いることにより，同様の機能を日本

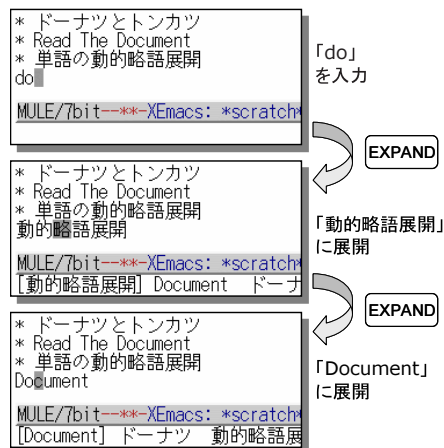


図 6 Nanashiki による動的略語展開
Fig. 6 Dynamic abbreviation expansion by Nanashiki.

語ファイル名においても実現可能である．たとえば `/tmp` ディレクトリ以下にあるファイル「情報処理学会.html」を閲覧する場合，UNIX では

```
% less /tmp/情報処理学会.html
```

というコマンドを入力する．

従来のファイル名の補完機能を用いて「情報処理学会.html」を入力するためには，

- (1) かな漢字変換システムの起動
- (2) 「jouhou」を「情報」に変換
- (3) かな漢字変換システムの終了
- (4) 「情報」を「情報処理学会.html」に展開

または

- (1) かな漢字変換システムの起動
- (2) 「jouhoushorigakkai」を「情報処理学会」に変換
- (3) かな漢字変換システムの終了
- (4) 「.html」の入力

というプロセスが必要である．対して，Nanashiki を用いれば

- (1) 「jo」を「情報処理学会.html」に展開のみでよく，ファイル名が日本語であっても効率的に入力ができる．

4. Nanashiki の予測入力への適用

我々は Emacs 用の POBox (図 7) に Nanashiki を組み込むことによって，文脈をとらえた予測入力を

`Tab` キー または `Ctrl` + `i` に割り当てられていることが多い．



図 7 Emacs 用 POBox (上:動作画面, 下:複数の変換エンジンの選択)

Fig. 7 POBox for Emacs (Top: Input of a word, Bottom: Selection of multiple conversion engines).

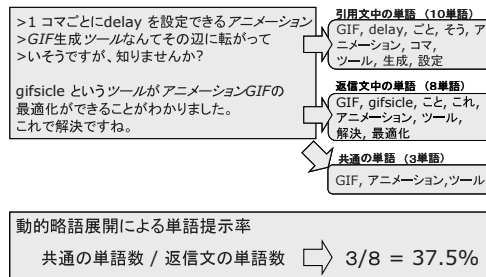


図 9 評価の例

Fig. 9 Example of the evaluation.

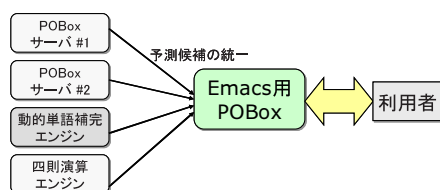


図 8 変換エンジンの統合

Fig. 8 Integration of conversion engines.

実現した .

Emacs 用の POBox は、複数の変換エンジンを統合的に扱う機能を持つ . そのため、通常の POBox 変換エンジンに加えて、さまざまな変換エンジンを組み込むことが可能である (図 8) . 動的略語展開による予測候補の追加の仕組みも、1 つの変換用エンジンとして実現した .

各変換エンジンが作成する単語候補の結合は、各変換エンジンに与えられた優先順位によって決定される . 優先順位の高い変換エンジンが予測した候補がより先頭に配置される . また重複した単語候補は、優先順位の高い単語候補を 1 つだけ残して、削除される . 本研究の動的略語展開による変換エンジンには、最大の優先順位が与えられている .

現在の実装では、動的略語展開による予測の対象は、利用者が編集中の文書のみとした . さらに多くの文書を対象とすることも可能であるが、処理速度を考慮して対象文書を限定した .

加えて、動的略語展開による単語の予測には制限時間を設けている . 単語の予測は制限時間を過ぎた時点で終了し、時間内に予測した単語のみを候補とする . 制限時間を設けることにより計算機の処理速度に左右されず、利用者にストレスを与えない時間内での予測が可能となる . なお、初期設定での制限時間は 0.2 秒とした . この 0.2 秒という長さは、我々が実際に本手法を使用して経験的に許容できると判断した時間で

ある .

5. 評価

動的略語展開を予測入力に応用する場合における予測入力の入力効率の向上性を評価するとともに、本手法の速度的な実用性について評価を行った .

5.1 予測入力の評価

動的略語展開による入力効率の向上を、メールのやりとりにおける相手のメッセージに含まれる単語の再利用率ととらえ、次の手順で測定した .

メッセージ分割 メールを、相手のメッセージ部分である引用部分と、利用者が作成した返信部分に分割する .

共通単語の抽出 分割したそれぞれのメッセージから、単語を抜き出し、共通して含まれている単語を取得する .

再利用率を計算 返信部分に含まれる単語のうち、引用部分と共通して含まれている単語の割合を、動的略語展開によって提示される単語の割合として求める .

名詞が文脈の情報を形成していると仮定し、本評価では調査する単語を名詞に限定した . 図 9 に評価の例を示す .

実際の評価は 6 つのメーリングリストから返信メールを抽出して行った . メーリングリストの種類はソフトウェア開発者間のコミュニケーション (開発者 1, 2), ソフトウェア利用者間のコミュニケーション (ユーザ 1, 2), 友人同士のジャンルを問わないコミュニケーション (友人 1, 2) を 2 つずつ選択した . その結果、どのメーリングリストもおおよそ平均 10% 前後の単語を動的略語展開によって提示できることが分かった . 図 10 に各メーリングリストの提示率の分布を箱髴図

たとえば、「最寄りの駅は自由が丘です」という文では、「の」、「は」、「が」、「です」といった助詞や助動詞は文脈の形成には寄与しない .

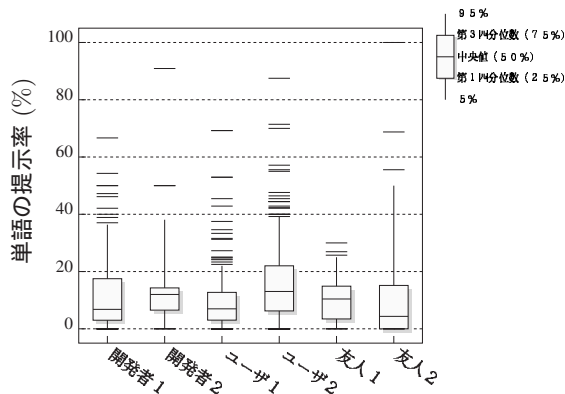


図 10 単語の提示率の分布

Fig. 10 Distribution of the prediction rate.

表 2 単語の提示率
Table 2 Prediction rate.

MLの種類	平均提示率	メール数
開発者 1	11.32%	212
開発者 2	13.73%	73
ユーザ 1	8.89%	363
ユーザ 2	15.40%	420
友人 1	10.31%	67
友人 2	11.29%	73
計	12.09%	1208

で、平均提示率を表で示す。

図 11 に示すように、高い提示率を示したメールは特定の話題について議論を行う形式に多く見られた。このような場面では動的略語展開による入力予測は効果的にはたらく。対して、低い提示率であったメールは、短い文章のメールであったり、話題が変化しているメールに多く見られた。この原因は、そもそも動的略語展開がとらえるべき文脈情報が引用メッセージの中に存在していないことである。表 3 が示すように、予測候補の提示率が低いメールには引用メッセージ中に含まれる単語数が少ないことが分かる。

加えて、表 4 および表 5 は図 11 の下線単語の入力に必要なストロークの比較である。表 4 は、対象単語が予測候補の第 3 候補以内となる条件下での比較であり、表 5 は、対象単語が第 1 候補となる条件下での比較である。また比較は、本論文の筆者によって十分に学習された環境で行った。

この比較は一例にすぎないが、本手法を用いると、利用者が普段入力しない単語であっても文脈を表す単語を従来方法の半分以下の打鍵数で入力可能であることがうかがえる。特に「MediaRecorder」などの固有名詞を入力する場面では、従来の方法ではすべての文

表 4 図 11 の下線単語入力に必要なストローク (第 3 候補以内)
Table 4 Stroke for underlined words in Fig.11 (within 3 candidates).

入力単語	従来方法	本手法
MediaRecorder	MediaRecorder	M
キャプチャ	kyap	k
スクリーン	sukuri	su

表 5 図 11 の下線単語入力に必要なストローク (第 1 候補)
Table 5 Stroke for underlined words in Fig.11 (first candidate).

入力単語	従来方法	本手法
MediaRecorder	MediaRecorder	Me
キャプチャ	kyap	ky
スクリーン	sukuri-n	su

表 6 評価環境

Table 6 Evaluation environment.

	Guest OS(評価環境)	Host OS
OS	Linux (2.4.19)	Windows XP
CPU	2365.84 BogoMIPS	Mobile Pentium III 1.20GHz
Memory	512MB	1GB

VMWare のバージョン: 3.2.0

VMWare の優先度: Normal

字を打鍵する必要があるのに比べ、本手法ならば 1 文字、2 文字程度を打鍵するだけで入力が可能である。

5.2 処理速度の評価

本論文で提案する手法の速度面での実用性を検証する。検証方法は、キー入力に対して 0.2 秒以内に取得が完了した変換候補の数と、その際に検索した文字数を計測することで行った。「検索した文字数」は、キー入力の位置からどれだけの範囲を本手法の対象とできるかの指針となる。たとえばこの「検索文字数」が 100 であれば、キー入力の位置から前後 50 文字に含まれる単語が、本手法で利用可能である。

評価に用いた文書は、本論文のドラフトのプレーンテキストである。キー入力の位置は、文書の文字数を 16 等分する 17 カ所で行い、同じ位置で 3 回ずつ計測を行った。評価環境としてノート PC の IBM ThinkPad X30 を用い、VMWare 上で GuestOS として動作している Linux によって行った(表 6)。キー入力がアルファベット 1 文字の場合の結果を表 7 に示す。

次にこの結果を基に、キー入力がアルファベット 1 文字の際に取得可能な候補の数と検索文字数の期待値を算出した。まず前提として、各変数を以下のように定義する。

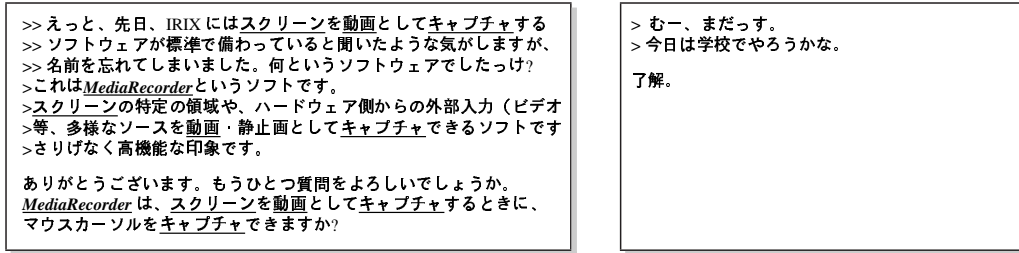


図 11 効果的な例（左図）と効果的ではない例（右図）
 Fig. 11 Effective example (left) and ineffective example (right).

表 3 提示率と引用メッセージに含まれる平均単語数の関係

Table 3 Relation between prediction rates and average word numbers in cited messages.

ML の種類	全体		提示率 5% 未満		提示率 5% 以上	
	メール数	平均単語数	メール数	平均単語数	メール数	平均単語数
開発者 1	212	23.3	80	14.4	132	28.7
開発者 2	73	20.6	11	16.2	62	21.4
ユーザ 1	363	17.4	135	9.8	228	21.8
ユーザ 2	420	21.4	90	11.8	330	24.0
友人 1	67	23.9	19	13.2	48	28.2
友人 2	73	13.0	33	9.2	40	16.1

表 7 1 文字入力時の結果

Table 7 Result for each one character.

入力	全単語数	候補数	検索文字数
a	87	5.41	243.57
b	64	4.31	476.67
c	43	2.37	280.37
d	82	3.47	281.45
e	26	0.08	478.25
f	38	4.22	474.31
g	71	3.02	337.73
h	159	2.90	185.65
i	70	4.65	263.73
j	65	4.39	395.08
k	222	2.69	183.69
l	4	1.27	1018.71
m	131	1.75	214.08
n	113	5.20	211.82
o	62	2.96	267.29
p	46	2.27	495.45
q	0	0.00	0.00
r	41	2.22	263.16
s	250	3.14	74.24
t	208	1.53	123.39
u	39	2.37	702.84
v	3	0.29	1936.41
w	23	3.63	491.35
x	1	0.00	1950.14
y	55	1.35	313.47
z	76	4.53	405.04

全行数: 313
 全文字数: 9104
 1 行あたりの平均文字数: 29.09

w_i = 入力文字 i から始まる全単語数

c_i = 入力文字 i に対する取得候補数

s_i = 入力文字 i に対する検索文字数

$$W = \sum_{i='a'..z'} w_i$$

たとえば入力文字 'a' に対しては, $w_a = 87$, $c_a = 5.41$, $s_a = 243.57$ である. また $W = 1979$ となる.

次に候補数と検索文字数の期待値を以下の式に基づいて算出する. つまり, 各文字の出現頻度を全単語と入力文字から始まる単語との割合から計算し, その出現頻度を用いて各期待値を算出している.

$$\text{候補数の期待値} = \sum_{i='a'..z'} (c_i w_i / W)$$

$$\text{検索文字数の期待値} = \sum_{i='a'..z'} (s_i w_i / W)$$

この式によって各期待値をもとめると,

$$\text{候補数の期待値} = 3.07$$

$$\text{検索文字数の期待値} = 252.97$$

となる. つまりアルファベットを 1 文字打鍵した場合, 動的略語展開によって予測される候補の数は約 3 候補であり, また約 253 文字分・8.7 行分の検索が行われることが期待される.

また同様にキー入力 'aa', 'ab', 'ac' から 'zz' までのアルファベット 2 文字分での計測を行ったところ

$$\text{候補数の期待値} = 3.61$$

$$\text{検索文字数の期待値} = 738.57$$

となった.

結論として, 1 文字入力する時点で得られる候補の数は 3 以上であり, また 2 文字入力すると約 25.4 行分の検索が行われるため, 十分実用的な動作速度であ

るといえる。

6. 議 論

文脈をとらえた予測入力は、メールなどのコミュニケーションに限らず、様々な場面で有効である。本章では、動的略語展開を利用する妥当性と有効性、および他の方法での文脈をとらえた予測入力について議論する。

6.1 動的略語展開の妥当性

本手法による単語の予測は、動的略語展開を利用者のキー入力ごとに毎回行っている。処理速度を考慮するならば、動的略語展開を毎回行うのではなく、予測単語の候補を、ファイルの読み込み時などに、あらかじめ作成しておく方が有利である。しかし、本手法では予測の柔軟性と精度を重視した結果、毎回動的略語展開を行う方法を採用した。

本手法では、毎回動的略語展開を行うため、たとえばコピーアンドペーストのような突発的な文書の変更に対しても、柔軟に対処可能である。加えて、動的略語展開ではカーソル位置と単語の距離を取得できるため、その距離に応じた優先順位の設定ができる。そのため、あらかじめ候補を作成する方法よりも予測の精度を上げることが可能である。

6.2 文書編集や校正に対する有効性

動的略語展開を利用した予測入力は、文書の編集や校正にも有効である。このような予測入力では、すでに文書中に存在する単語を優先的に扱うため、編集時や校正時に誤りやすい単語の揺れを防ぐことができる。たとえば「行く」の送り仮名は、人によっては「行なう」と記述される。また「かたかな」を「カタカナ」や「片仮名」と記述することもよくある。これらの文体は文書内で統一すべきであるが、他人の文書の校正や過去に作成した文書の修正を行う場面では、単語の揺れが起きやすい。このような単語の揺れを防ぐ面からも、動的略語展開を利用した予測入力は有効である。

6.3 他の方法での文脈をとらえた予測入力

文脈をとらえた予測入力の実現には、動的略語展開を応用する方法以外にも、いくつか考えられる。これらの方法と動的略語展開を組み合わせることで、より効果的な単語の予測が期待できる。

我々が開発し提案した文書蓄積システム Kukura「句倉」を適用すれば、より広範囲の文脈をとらえた予測入力を実現できる¹⁰⁾。Kukuraは、利用者が扱ったあらゆる文書を保存・活用するシステムである。Kukuraを用いることにより、たとえば、ウェブブラウザを参照しながらメールを作成する場合において、ウェブ

ブラウザに表示されている単語をメールの作成で候補として提示することが可能になる。また、文書検索システムなどと Kukura を組み合わせることにより、現在編集中の文書と関連性の高い文書から予測候補を作成するといった方法も考えられる。

加えて、朝や夜といった時間、職場や家といった環境、論文やカジュアルなコミュニケーションといった目的などに応じて予測方法を切り替えるといった手法も考えられる。

7. 結 論

我々は日本語の予測入力の新たな手法として、文書の文脈の重要性に着目した。文脈に着目することにより、メールのやりとりなどのコミュニケーションにおける入力において、効率的な予測が可能となる。我々は、動的略語展開を利用した文脈をとらえた予測入力を実現し、その有効性を実証した。今後、メールやチャットなどの電子的なコミュニケーションがより日常的になるにつれて、本論文で提案した予測入力システムは、ますます重要になると考えている。

参 考 文 献

- 1) Darragh, J. J., Witten, I. H. and James, M. L.: The Reactive Keyboard: A Predictive Typing Aid, *IEEE Computer*, Vol. 23, No. 11, pp. 41-49 (1990).
- 2) Masui, T.: An Efficient Text Input Method for Pen-based Computers, *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '98)*, Addison-Wesley, pp. 328-335 (1998).
- 3) 市村由美, 齋藤佳美, 木村和広, 平川秀樹: 予測に基づく入力支援機能を備えたかな漢字変換システムの開発, 情報処理学会研究会報告 自然言語処理, Vol. 129, No. 10, pp. 63-70 (1999).
- 4) 田中久美子, 犬塚祐介, 武市正人: 携帯電話における日本語入力-子音だけで日本語が入力できるか, 情報処理学会論文誌, Vol. 43, No. 10, pp. 3087-3096 (2002).
- 5) 福島俊一, 山田洋志: ペン予測入力インタフェースとその手書き操作削減効果, 情報処理学会論文誌, Vol. 37, No. 1, pp. 23-30 (1996).
- 6) 富士通株式会社: Japanist (2000). <http://software.fujitsu.com/jp/japanist/>.
- 7) 井田昌之, 亀井信義: Emacs 解剖学 入力の補完, *bit*, Vol. 29, No. 2, pp. 85-95 (1997).
- 8) 高林哲, 小松弘幸, 増井俊之: Migemo: 日本語のインクリメンタル検索, 情報処理学会論文誌, Vol. 43, No. 12, pp. 3698-3705 (2002).
- 9) 松本裕治: 形態素解析システム「茶釜」, 情報処

理, Vol. 41, No. 11, pp. 1208–1214 (2000).

- 10) 小松弘幸, 高林哲, 増井俊之: 文書蓄積システム Kukura を用いた予測入力, 日本ソフトウェア科学会 WISS2002, pp. 43–47 (2002).

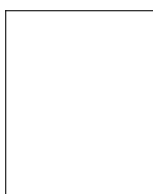
(平成 14 年 4 月 14 日受付)

(平成 14 年 9 月 5 日採録)



高林 哲 (学生会員)

1976 年生。1999 年愛知大学経営学部経営学科卒業。2001 年奈良先端科学技術大学院大学博士前期課程修了。同年同大学院博士後期課程入学。現在、独立行政法人産業技術総合研究所勤務。2002 年情報処理学会山下記念研究賞受賞。



小松 弘幸 (学生会員)

1976 年生。2000 年東京理科大学卒業。2002 年東京工業大学大学院情報理工学研究科博士前期課程修了。同年同大学院博士後期課程入学。察するインタフェースに興味を持つ。



増井 俊之 (正会員)

1984 年東京大学大学院工学系研究科電子工学専攻修士課程修了。ユーザインタフェース関連の研究に従事。工学博士。現在、独立行政法人産業技術総合研究所勤務。携帯端末のインタフェース、情報検索、富豪的プログラミングに興味を持つ。