適応 / 予測型テキスト編集システム

A Simple Approach to Adaptive Text Editors

増井 俊之*

Summary. We propose a simple adaptive predictive interface for text editing tasks. A text editor can predict the next user input from various information available, such as the user's repetitive operations, a dictionary of frequently-used wors, preveously-input text strings, etc. Using a text editor with such a predictive feature, a user can ask the system to predict his next operation and select the appropriate one from the candidates shown by the system. If the system can also keep track of the user's selections and know the user's preferences, it can gradually adapt itself to the user and show more appropriate candidates next time. We implemented five prediction schemes on GNU Emacs, and also implemented one simple adaptation scheme for ordering the prediction schemes. Empirical results show that adaptive predictive interface shows higher usability than non-adaptive interface.

1 はじめに

操作履歴と編集中のテキスト情報を利用して適応的に次の操作の予測を行なう文書編集システムを提案する．本システムでは操作の履歴と現在のコンテキスト情報のみから予測を行ない，このとき， 1) 各種の予測手法を併用する 2) 予測手法を適応的に選択する 3) 適応手法やパラメタをユーザが修正可能とする ことにより，単純で効果的な適応型予測インタフェースを構築できることを示す．以前に提案した統合的繰り返し予測手法を拡張して適応型とすることにより，さらに有用な予測を適応的に使うことができるようになった．

2 適応型インタフェース

システムが個々のユーザの特徴を何らかの方法で検出してそれに応じて挙動を変える，いわゆる適応型 (adaptive) システムが近年注目を集めている．適応型インタフェースシステムは，繰り返し使用するうちにシステムが使いやすくなったり，個人の特性に対応して操作方法やシステムの挙動が変わったり，適応したシステムを再利用できたりするという利点を持っている．

「適応」という言葉は広い範囲のインタフェースにおいて使用されているが，大きく**繰り返し操作の支援**と**ユーザの特性への対処**のふたつに分類することができる．

* Toshiyuki Masui, シャープ株式会社 ソフトウェア研究所

2.1 繰り返し操作の支援

ユーザが似た操作を何度も実行しているとき，システムはそれを検出して以下のような各種の方法でその作業を支援することができる．

- マクロ作成支援

  似た操作を何度も繰り返す場合は，繰り返される操作列をマクロとして登録しておくのが便利であるため，ユーザの操作列からマクロ定義可能な共通部分を取り出すシステムが提案されている． IBM-PC 上の KeyWatch システム [13] は，同じキーストローク列の繰り返しを発見すると音などでユーザにそれを知らせ，マクロとして再実行可能にする． Dynamic Macro(**DM**)[12] では，テキストエディタにおいてユーザが「繰り返し実行キー」を押すと，システムがそれまでの操作履歴から繰り返しパタンを検出してマクロとして登録 / 実行する． Eager[3] は， Macintosh の HyperCard でユーザが似た操作を繰り返した場合システムがそれを検出し，以後もその操作を繰り返すかどうかをユーザに問いあわせる．

  あらかじめ用意されたアプリケーション知識を利用してさらに高レベルの繰り返し操作を検出するシステムもある． UIDE[15] は，アプリケーション知識を表現する階層的データ構造を参照することによりユーザ操作列の中でマクロ化可能な部分を抽出する．また Flexcel[16] は， Excel の知識をもつエキスパートシステムがユーザの Excel に対する操作をモニタすることにより，マクロとして定義可能な操作列を検出する．

- 選択候補の並べかえ

  候補をリストにして提示するメニュー型インタフェースや，かな漢字変換システムのように候補を順番に提示するシステムにおいては，よく使われる候補をプルダウンメニューの上部に配置したり，候補列の最初の方に持ってきたりすることが有効である． SplitMenu[14] は，プルダウンメニュー中の項目のうちよく使われるものを他の項目と分離してメニューの最上部に表示されるようにしたものである．また Greenberg らは階層的電話帳メニューにおいて，よく選択される名前がメニューの階層の上の方に出現するようにした場合の効果について報告している [6]．報告されている． Kuhme の Action Prompting システム [8] は，現在のコンテキスト及びユーザの操作履歴から判断して次に選択されそうなメニュー項目のリストを選び，それを常時表示のメニューとして通常のメニューと別の位置に表示するようになっている．

- 予測される操作の提示

  Reactive Keyboard システム [5] は， UNIX のシェルのようなコマンド言語インタフェースにおいて，ユーザの操作履歴の統計情報にもとづいてユーザの次の操作を予測し提示する．

- 知的繰り返し作業の支援

  Maes はメモリベースの機械学習機構を用いて多くの実例により「エージェント」を学習させることにより，エージェントがスケジュール調整や電子メールの仕分けを自動的に実行できるようにする枠組を提案している [10].

2.2 ユーザの特性への対処

繰り返し作業への適応は一時的なものが多いが，ユーザの保有する各種の特性に合致するようにシステムを適応させるための手法も各種提案されている．

- 典型的ユーザモデルとのマッチング

  ユーザの知識や技能レベルに応じてオンラインヘルプやインタフェースを変化させる試みが行なわれている． Benyon のシステム [1] ではシステムがユーザの知識や能力を判断してコマンド言語インタフェースとメニューを使うインタフェースを切り換えるようになっている． UIDE[15] では，アプリケーション知識を示すデータ構造の部分構造がユーザの知識を示しているという考えにもとづき，ユーザが実行したことのない操作に対して重点的にガイダンスを行なうようになっている．また Flexcel[16] では，ユーザの操作履歴からそのユーザの性質を判断することによりユーザに適応した動作をするようになっている．

- パラメタのチューニング

  文字認識や音声認識などの分野では，個人の字や声などの特徴を抽出するためのパラメタをシステムが計算することが広く行なわれている．

- ユーザの嗜好の抽出

  ユーザの嗜好や感性のような数値化しにくい特性に対しても適応を行なうための枠組が提案されている．前述の Maes の学習エージェントの考え方に基づいたスケジュール調整システム [7] では，エージェントは会議の時間の調整を通じてユーザの時間の好みや会議の相手の重要度を学習していく．また筆者は，個人の美的感性を表現するようなグラフ配置の評価関数を遺伝的プログラミングの手法を用いて抽出する手法を提案している [11].

3 適応型インタフェースの問題点

2節で述べたように，適応型インタフェースの目標とする理想は高いものの，実際の成功例は少ないようである．適応型インタフェースがうまくいかない理由のいくつかを以下に述べる．

- 「バンド幅」問題

  適応型インタフェースシステムはユーザの操作履歴をたよりに各種の判断を行なうため，ユーザとシステムの間のインタラクションが少ないときは有効な判断を行なうことができない．例えば自動車を運転中しているユーザ (運転者) が少しだけハンドルを右に切ったような場合，システム (自動車) がその理由を解析することは不可能に近いので，適応もまた不可能である．

- モデルの問題

  適応型インタフェースの構築にはアプリケーションのモデルとユーザのモデルがよく使用されるが，モデルの構築 / 使用には問題点が多い．アプリケーションモデルは詳細なアプリケーション知識を持つ必要があるが，複雑なアプリケーションの知識を効果的に表現することは困難であるし，アプリケーション作成にそれだけ手間がかかることになる．またユーザモデルとしてユーザの各種の特性を数

値表現した値の組が使われることが多いが，このようなデータは適応に対する有効性 / 正当性が疑問である．モデルの作成のためにはユーザへの質問に対する回答または操作履歴が使用されるが，両者ともに信用のできるものではない．

- 適応に対する不安感

  システムの適応がユーザの知らないうちに勝手に実行されるとユーザは不安を覚えることが多いのでこのような仕様は避けなければならないが，ユーザに確認を求めた後で適応動作を行なうようにしたとしても，ユーザはシステムの挙動や操作性が以前と変わることに抵抗があるはずである．また，何故システムが適応しようとしたかについてユーザが理解していない場合には不安な感じがするであろうし，適応のしくみをユーザが理解していたとしても，そのしくみをユーザが操作可能でなければやはり不安を感じたりいらだちを覚えたりするかもしれない．

- 適応の有効性の疑問

  適応戦略が優れたものであったとしても，適応による効果が大きくなければ適応の意味は無い．ユーザや状況によりシステムの動作を変えることが本当に有効な場合でないと駄目である．適応などしなくても万人が使いやすいインタフェースがあればそれが一番良いのだから，そもそも適応型インタフェースに関して疑問視する考えもある．

4 適応 / 予測型テキスト編集システム

前節で述べたように適応型インタフェースの成功例は今のところ少ないが，予測インタフェースと組み合わせることにより効果的な適応型インタフェースを構築することが可能であると考えられる．本節では，統合的繰り返し予測手法を適応的に拡張した文書編集システムを提案する．

4.1 予測型テキスト編集システム

4.1.1 統合的繰り返し予測手法

Dynamic Macro(**DM**) [12] は，「繰り返し実行キー」 (以降REPEATと表記) によるユーザの指示により文書編集操作の繰返しをシステムが検出して繰り返しパタンをマクロとして登録し実行する機能である．またこれに加えて「予測実行キー」 (以降PREDICTと表記) により予測手法や候補を切り換えることにより，様々な予測方式を選択し繰り返し実行させることができるというものである．

図 1にREPEATとPREDICTを併用して予測手法を切り換える例を示す．TAB TAB a b c RET TAB TABを入力した後でREPEATを入力すると，TABが最後に 2 度続いて入力されているために次の入力としてTABが予測され，ここでさらにREPEATを入力するとTABが繰り返されるが，この予測がユーザの期待に反していた場合この状態でPREDICTを押すと，別の候補であるa b c RETが予測されるようになる．これがユーザの期待と一致する場合は，この後再びREPEATを押すことによりこの予測結果を繰り返させることができる．

PREDICTは単体で用いると，辞書による completion などを用いて次の入力の予測を行なう． completion とは，現在のカーソル位置の直前の文字列が辞書中の文字列の先頭

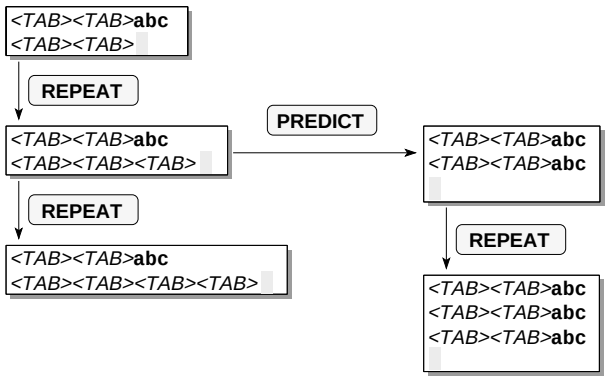

図 **1.** REPEAT による予測と PREDICT による候補の切り替え

部分文字列になっているとき，カーソル位置以降に残りの部分が続くことを予測するものである． completion による予測の様子を図 2に示す．

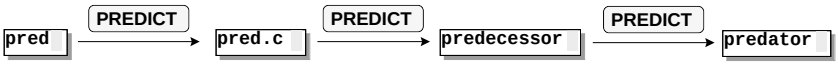

図 **2.** PREDICT による completion 予測

4.1.2 その他の予測方式

予測手法として **DM** と completion 以外に以下のものを使用する．

- Dynamic Abbreviation (`dabbrev`)

  completion と同様にカーソル直前の文字列からその先の文字列を予測するが，文書の前の方に既に出現している文字列を辞書として使用する．例えば文書の前の方に “`abbreviation`” という文字列が存在するとき， “`abb`” という文字列から “`abbreviation`” を予測する．空白記号やデリミタ記号を文字列の区切りとみなす．

- 2 度以上出現した文字列の使用 (twice)

  `dabbrev` では空白やデリミタ記号を含む文字列は予測に使用できなかったが，それらを含む文字列でも文書中に 2 度以上出現していれば completion の辞書として使用できるようにしている．例えば “SHARP␣`Corp.`” という文字列が 2 度以上文書中に出現しているとき， “SHA” から “SHARP␣`Corp.`” を予測する．

- 操作履歴中に 2 度以上出現した操作列の使用 (twice-history)

  これは twice と **DM** を組み合わせたようなものである．同じ操作列が 2 度以上操作履歴中に出現しており，かつその操作列の最初の部分列を実行した直後であれば，残りの部分が実行されると予測する． **DM** では操作を 2 度繰り返した直後でないとその操作を再実行させることはできなかったが， twice-history による予測機能を使うと，以前に何度か実行されたことのある操作列であればその操作列の最初の部分を実行した後で残りの部分を予測させることができる．

それぞれの予測手法にもとづいて候補リストが別個に作成され，それらがPREDICTを押す度に順番に提示される．ひとつの予測手法による候補が全て試された後で次の予測手法による候補が試される．例えば図 3の状態において次に来る文字列を予測させたとすると，候補リストは図 4のようになる．ここでは completion 辞書に “adaptation” 及び “adapt” が入っていたと仮定している．

```
An adaptive system should also be
adaptable, since not only the
system, but also the user, should
be able to make the system ada
```

図 **3.** 予測前の状態

予測手法	候補
dabbrev	adaptable adaptive
completion	adaptation adapt
twice	adapt

図 **4.** 予測手法と候補

以上のような各種の予測方式の併用により，[12] のシステムでは不可能だった予測も可能となっているが，それだけ複雑になっているため場合に応じてうまく予測手法を使いわけることが必要になる．そこで適応的にこれらの予測を利用することが有効になる．

4.2 適応インタフェースと予測インタフェースの関連

2節で述べたように，適応型インタフェースは予測インタフェースと組みあわせて用いられることが多い．特に，予測が一度で的中する確率が高くない場合は，複数の予測候補の中から目的の候補を選択する必要があるので，これを適応的に行なうことが効果がある．例えば，かな漢字変換の的中率は必ずしも高くないが，直前に選択した候補を次回の最初の候補とするといった適応方策により使い勝手が向上する．

GNU Emacs における各種の予測手法及びその確信度と間違いに対する対処法を図 5に示す．下の列ほど予測の的中度が低いと考えられ，それだけ適応型インタフェースを採用するメリットが大きいと考えられる．実際，適応機能が最も役にたっているものはかな漢字変換の候補並びかえ機能であると思われる．罫線引き機能 [17] とは矢印キーにより罫線を引きながらカーソルを移動させる機能であり，現在のコンテキスト (カーソルの下の罫線の方向) 及び前の操作 (罫線描画方向) から次の動作を決定しているため一種の予測インタフェースとみることもできるが，予測がはずれることはほとんど無いためそのようには認識されていないと思われる．

	予測に使う情報	確信度	誤り対策
一般コマンド	(なし)	高	undo
罫線引き	カーソル直下の文字及び操作履歴	高	undo
REPEAT	操作履歴	中	PREDICT
dabbrev	前の文字列	低	dabbrev キー
PREDICT	前の文字列など	低	PREDICT
漢字変換キー	前の文字列	低	変換キー

図 **5.** 各種予測インタフェースと予測の確信度

4.3 予測機能と適応機能の組み合わせ

4.1.2節で述べた予測手法を適応的に使うための手法を以下に述べる． 3節で適応型インタフェースの問題点について述べたが，これらを避けるために以下の戦略を採用する．

- 単純な適応戦略をとる

  かな漢字変換における適応型候補提示手法にみられるように，単純な予測手法を用いているときは複雑なアプリケーションモデル / ユーザモデルを使わなくても実用的な適応型インタフェースを構築可能であるため，できるだけ単純な適応戦略を採用する．

- 適応戦略を確認 / 修正可能とする

  どのような適応が起こっているのかをユーザが常に把握できるようにする．また適応方式やパラメタをユーザが修正可能とする．単純な適応戦略を採用していればこれらは容易に実現できる．

- システムが勝手に適応しないようにする

  ユーザの指示なしにシステムが挙動を変えないようにする．

具体的には，前回選択した方式や候補を次回優先的に使用する / 何度も選択した文字列は辞書登録する / 適応動作をユーザに提示する / カスタマイズを可能にする / という方式を採用する．これらのそれぞれについて以下に解説する．

4.3.1 前回選択した方式と候補の優先的使用

図 4に示したように，PREDICTにより予測を行なわせたときの候補は一般に複数存在する．この例において正しい予測は completion 予測による “adapt” であるが，図 4の順番で候補が示されたとすると正解に達するまでにPREDICTは 4 回押されなければならないことになる．ここで “adapt” が確定されると， completion を予測手法リストの先頭に移動しかつ “adapt” を completion 辞書リストの先頭に移動することにより，前回選択した方式 / 候補の優先的使用を指示することができる．その後前回と同様の予測を実行させると， 1 回目のPREDICTで候補 “adapt” が提示される．

4.3.2 何度も選択した文字列の辞書登録

dabbrev 予測や twice 予測により何度も確定された文字列は，頻繁に使用される有用な文字列と考えられるため completion 辞書に登録することにする．今回は 2 度確定し

たものは自動的に辞書に登録するようにしている．

4.3.3 適応動作のユーザへの提示

インタフェースが効果的に使われるためには，システムの適応動作についてユーザが明確に把握していることが必要である [9]．図 4の例では，システムは予測候補文字列として "adaptable", "adaptive", "adaptation", "adapt", "adapt" の 5 個を生成しているが， 4 個目の "adapt" は completion 辞書に含まれていた単語であるのに対し， 5 個目の "adapt" は "adaptable" と "adaptive" の共通文字列という意味しか持っていない．この事実をユーザが把握していればユーザは 4 個目の "adapt" の方が 5 個目のものより重要であることを認識して作業を続けることができるが，把握していなければこれらの区別をすることができない．ユーザが適応 / 予測方式を把握しながら作業ができるようにするため，現在の予測方式が常にユーザに提示されるようにしている．

4.3.4 カスタマイズ機能

個人用 completion 辞書・ completion 辞書に入れるための確定回数・適応戦略の提示の有無・ twice 予測に使用するプレフィクスの長さなどがカスタマイズ可能になっている．

5 実験及び評価

以上のような機能を組み込んだ予測 / 適応 Emacs を現在使用し評価中である．実装後日が浅いため今のところ評価は充分でないが， twice などの新しい予測方式及び適応手法を導入したことにより以前のシステムでは不可能だった予測を行なうことができるようになった点と，よく使う予測手法や候補が先に提示されるようになった点は便利になったと感じている．反面 twice や twice-history は予想外の候補を提示することも多く，場合によってはこれらの予測手法を全く導入しない方が混乱が少なくなることも考えられる．これらの予測手法の改良及び適応手法の改良が必要と思われる．

6 議論

6.1 適応型インタフェースのあるべき姿

3章で述べたように，効果的な適応型インタフェースの構築はむずかしい．各種の仕事，遊び，インタフェースなどにおいてユーザがとる行動を，判断基準の量 / 判断基準の明確さ / 判断に必要な思考の複雑さ / 結果行動の量 で分類したものを図 6に示す．暗い色で表現された領域ほど予測を行ないやすいことを示している．判断基準が明確かつ単純なときに予測が有効であるが，このような場合には今回紹介したような単純な適応手法が有効であると考えられる．また，判断のための思考が複雑でもその結果とられる行動が単純な場合や，判断基準がはっきりしなくても単純な基準から単純な行動がとられる場合には行動を予測する余地があると思われる．これらの場合は予測の的中率が低くなるため学習による適応手法 [10] や進化的手法 [11] により予測手法を修正していくことがよいであろう．

図 **6.** 仕事の分類と予測 / 適応の可能性

6.2 ユーザモデルの使用

本システムはユーザモデルを全く作らず操作履歴とコンテキスト情報のみから予測 / 適応を行なっている．[12] のシステムにおいても同様の手法で成功しているし，[2] の適応型 UNIX コマンドインタフェースにおける実験でも同様の結論が得られている．これらのように単純な予測を行なうシステムにおいてはモデルを使用しない方が有効と考えられるが，複雑なシステムにおいても同じかどうかは議論の余地があるだろう．

7 結論

操作履歴とコンテキスト情報のみを利用して適応的に次の操作の予測を行なう文書編集システムを提案し評価した．単純な適応機能しか用いていないにもかかわらず，適応機能を用いない予測インタフェースに比べ効率の良い予測が行なわれることが示された．今後本方式の評価を続けるとともにさらに効果的な予測手法について実験していく予定である．

8 謝辞

DM をはじめとする予測インタフェースの実装，評価にあたり数多くの有益な助言をいただいた東京大学の中山健氏に感謝します．

参考文献

[1] Devid Benyon and Dianne Murray. Developing adaptive systems to fit individual aptitudes. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp. 115–

121. ACM Press, January 1993.

[2] Daniel Crow and Barbara Smith. The role of built-in knowledge in adaptive interface systems. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp. 97–104. ACM Press, January 1993.

[3] Allen Cypher. Eager: Programming repetitive tasks by example. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pp. 33–39. Addison-Wesley, April 1991. also in [4].

[4] Allen Cypher, editor. *Watch What I Do – Programming by Demonstration*. The MIT Press, Cambridge, MA 02142, 1993.

[5] John J. Darragh, Ian H. Witten, and Mark L. James. The Reactive Keyboard: A predictive typing aid. *IEEE Computer*, Vol. 23, No. 11, pp. 41–49, November 1990.

[6] Saul Greenberg and Ian H. Witten. Adaptive personalized interfaces - a question of viability. *Behaviour and Information Technology*, Vol. 4, No. 1, pp. 31–35, 1984.

[7] Robyn Kozierok and Pattie Maes. A learning interface agent for scheduling meetings. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp. 81–88. ACM Press, January 1993.

[8] Thomas Kuhme. A user-centered approach to adaptive interfaces. *Knowledge-Based Systems*, Vol. 6, No. 4, pp. 239–248, December 1993.

[9] Thomas Kuhme. A user-centered approach to adaptive interfaces. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp. 243–245. ACM Press, January 1993.

[10] Pattie Maes. Learning interface agents. In *Proceedings of the 1994 Friend21 International Symposium on Next Generation Human Interface*, February 1994.

[11] Toshiyuki Masui. Evolutionary learning of graph layout constraints from examples. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'94)*. ACM Press, November 1994. to appear.

[12] Toshiyuki Masui and Ken Nakayama. Repeat and predict – two keys to efficient text editing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, pp. 118–123. Addison-Wesley, April 1994.

[13] Micro Logic Corp., POB 70, Hackensack, NJ 07602. *KeyWatch*, 1990.

[14] Andrew Sears and Ben Shneiderman. Split menus: Effectively using selection frequency to organize menus. *ACM Transactions on Computer-Human Interaction*, Vol. 1, No. 1, pp. 27–51, March 1994.

[15] Piyawadee Sukaviriya and James D. Foley. Supporting adaptive interfaces in a knowledge-based user interface environment. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp. 107–113. ACM Press, January 1993.

[16] Chritoph G. Thomas and Mete Krogsaeter. An adaptive environment for the user interface of Excel. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp. 123–130. ACM Press, January 1993.

[17] 増井俊之. `keisen.el` プログラム, July 1991. `etlport.etl.go.jp` の Nemacs contrib ディレクトリより入手可能.