相違度を用いた補完機能の実現

2K−2

○川邊　滋　　小山　裕徳
東京電機大学

1．はじめに

ＵＮＩＸのｃｓｈなどには、過去に実行したコマンドを記憶するヒストリ機能がある。この機能により、ユーザはヒストリ番号と呼ばれる数字か、先頭の数文字を入力することで、コマンドを再実行することができる。ヒストリ番号による方法は、番号とコマンドの対応関係を記憶することがユーザの負担となるが、数文字から再生する方法は、このような記憶が不要なため、ユーザに負担をかけずにキー入力数を減少させることができる。この数文字から再生する機能は補完機能とも呼ばれ、句切り文字を拡張すればコマンド単位の補完だけでなく、単語単位の補完機能も実現できるようになる[1]。

しかし、従来の補完機能ではユーザが入力した文字列が、目的の文字列の先頭部分と一致していなければならず、もし入力文字列に誤りがあると、再生に失敗して、意図した補完ができない。この問題を解決するため、入力文字列と目的の文字列の間に「相違度」という尺度を導入し、これにもとづく相違度補完を実現した。

2．相違度の定義

キー入力中にユーザが起こす誤りは次の４種類のいずれかに分類することができる。

(1)過多　余計な文字を挿入してしまう誤り
（例：ls→las）

(2)不足　必要な文字を欠落してしまう誤り
（例：mkdir→mdir）

(3)誤字　誤った文字を入力してしまう誤り
（例：mail→msil）

(4)置換　前後の文字を入れ違う誤り
（例：cat→cta）

ユーザが起こした入力誤りの程度を相違度と定義する。ユーザの入力した文字列ｘに対する目的の文字列ｙの相違度ｄ（ｘ，ｙ）は次の式で与えられる。

ｄ（ｘ，ｙ）＝ｎ　（ｎ≧０）

ここでｎは、ｘをｙの先頭部分に一致させるために必要な最小限の操作量であり、(1)から(4)に対応する変換操作の回数である。

相違度ｄは次のように求める。文字列ｘ、ｙに対し、先頭の文字から１文字づつ比較して、(1)(2)に相当する誤りを判定する。(1)(2)が同時に発生すれば、(4)であると判定する。そして、単に字が異なる場合は(3)であると判定する。これを文字列ｘの終りまで順次繰り返し、誤りと判定された回数の総和をｄとする。

定義から明かなように、ユーザが正しく入力した場合（ｘがｙの先頭部分に一致するとき）には相違度が０になる。また、相違度がｎであるときは、(1)から(4)に対応する変換操作をｘにｎ回施すと、目的の文字列ｙの先頭部分に一致させることができることを意味している。

3．相違度補完の原理

ユーザが文字列Ｒを入力しようとしたとする。このとき、実際に入力した文字列をＩ、Ｒにより補完される目的の文字列をＣとする。ＲはＣの先頭部分に一致しており、相違度は０である。もし、ＩとＲの相違度がｎであれば、ＲとＣの関係からＩとＣの相違度もｎとなる（図１）。ところで、Ｃはデータバッファ上に登録された文字列に限定されている。そのため、ＤＢ上の全ての文字列に対してＩとの相違度を求めると、Ｒがわからなくてもを推定することが可能である。なぜなら、Ｉに誤りが無ければ相違度０の文字列がＣになり、１つ誤りがあるならば相違度１の文字列がＣになるためである。ユーザが意図的に入力を誤ることは、通常考えられないので、補完すべき目的の文字列は、相違度が一番小さい文字列であると考えることができる。コマンドレベルとアプリケーションレベルでは文字列に対する句切り文字が異なるだけであるから考え方に差は無く、同一の方式で相違度補完が実現できる。

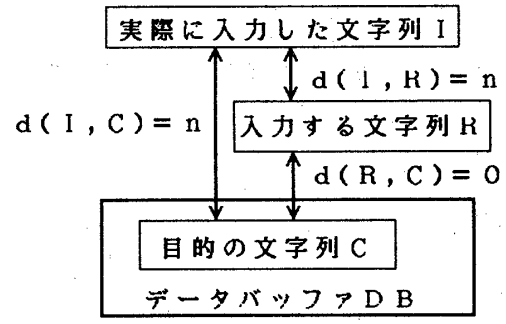

図１　相違度の関連図

4．試作システム

コマンドレベルは次の事項に重点をおき、図２に示した構造となっている。

・Line editor
補完された文字列がユーザの意図したものでないときや、意図的に書き換える場合を考え、スクリーンエディタなみの編集機能を付加する。

Implementation of completion function based on the degree of difference
Shigeru KAWABE , Hironori KOYAMA
Tokyo Denki University

・Buffer
補完できる文字列が限定されることは汎用性に欠けるため、キー入力を監視してデータバッファを逐次更新するようにする。このとき、バッファ内の冗長性を極力排するために、文字列が重複しないように配慮する。また、最近入力された情報ほど優先度を高くする。

・Trigger
補完機能はユーザが決めたトリガを押すことによって起動する。ユーザの意思によって補完機能を使い分けることができるため、従来のユーザの環境をほとんど損なわない。また、連続してトリガを押すことにより、優先度の低い次候補などが選択できるようにする。

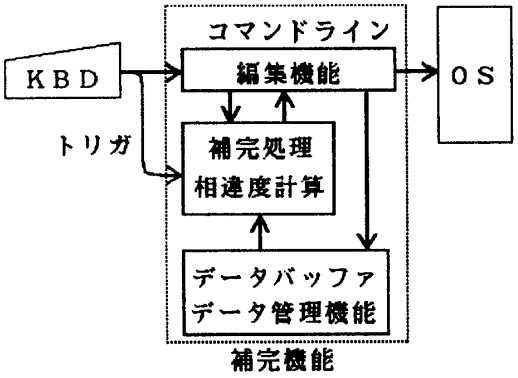

図2　試作システム構造図
（コマンドレベル）

コマンドレベルでは、入力した文字列をコマンドライン上で自由に編集することができる。トリガが押されると、コマンドラインとデータバッファとで相違度が計算され、コマンドラインは相違度の一番小さい補完候補に書き換えられる。改行を入力すると、コマンドライン上の文字列はデータバッファを更新し、次いでOSへ転送されてコマンドとして実行される。

5．検証

相違度補完はユーザの意思により起動するため、数量的な検証が難しく、主観評価などによる心理的な検証によらざるをえない。しかし、相違度については、ヒット率で有為性が確かめられるため、そのための実験を行った。

実験は、相違度1を持たせた2から5文字の任意の文字列を入力文字として、目的の文字列が何回目のトリガによってヒットするか、その割合を調べたものである。補完の母集団となるデータバッファには、試作システムの動作するUNIX(SONY NEWS NWS-1720)上の全ディレクトリ上にある実行型ファイル名（約500）をアルファベット順に格納した。試行回数は各々の条件に対し1万回とし合計16万回行った。結果を累積グラフにして図3、図4に示す。なお、実験に要した時間は、およそ4200秒弱であった。1回の試行に要する時間は25[ms]程度となるから、相違度補完は実用上問題なく利用できることがわかる。

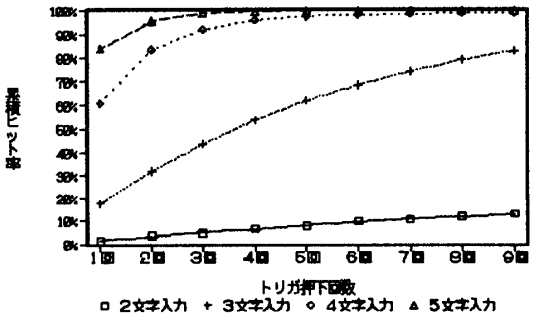

図3　入力文字数別補完ヒット率

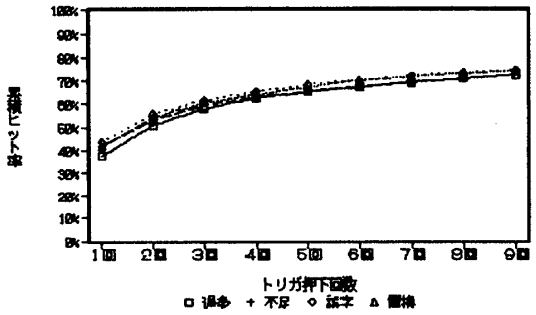

図4　誤入力別補完ヒット率

図3では、入力文字数に対する誤入力数（相違度）の比が小さいほどヒット率が上がっている。相違度が1である場合、入力文字が2文字ならば致命的であるし、4文字以上ならばあまり問題とはなっていないことがわかる。図4では、誤入力の種別に対する相違度補完に差があまり見られない。このことから、相違度の定義には矛盾がないことが確かめられた。

今回の実験では、データバッファ内の要素の並びが入力文字列に対して関連性がなく、しかも固定されているため、誤入力の割合が単純にヒット率に影響したものと思われる。実際に相違度補完を使用する場合には、データバッファの内容が入力文字列で逐次更新されるので、実験で得られた結果以上にヒット率が上がることは確実である。

6．おわりに

相違度補完の有為性は確かめられた。この試作システムでは、コマンドレベルがヒストリ置換、アプリケーションレベルがフロントエンドプロセッサとして機能する。コマンドレベルでの実用性は、経験的に十分確められているが、アプリケーションレベルは、画面制御などハードウェアへの依存度が高いため、カスタマイズ機能を取り入れるなどの対策が必要と思われる。

参考文献

[1] 川邊、小山「shellにおける補完機能の実現」情報処理学会第41回全国大会論文集(5-82)