子音情報のコード化と個人の誤り履歴を利用した電子英和辞書曖昧検索システムの開発

静岡大学大学院 理工学研究科 システム工学専攻
若林研究室 5083-0177 永谷真一郎

1 はじめに

電子辞書の検索にはいくつかの方法が提案され，利用されているが，ほとんどが入力文字列中の文字を使って，その組み合わせで得られる文字列を手がかりにして検索するものである．このような電子英和辞書ツールを日本人が使うことを考えた場合，英語に不慣れな日本人ならばローマ字表記に近い綴りをする傾向にあるといったことも考えられる．したがって，そのような利用者の特性を考慮に入れた検索を実現することが望まれる．

本研究では，文字列の子音情報を音声的にコード化する soundex[1] を利用して，英和辞書を音声的に近いと思われる項目について曖昧検索する特徴を持った辞書検索ツールの開発を行い，UNIX 上てのスペルチェッカ ispell と開発したツールとの比較を行い，その有効性を検証する．

2 従来の電子辞書の問題点

電子辞書の検索方法には色々な種類が存在する．利用者によって多くの知識を持たずとも利用できるパターン検索やワイルドカード検索などは良く利用されるが，正規表現のような複雑なものはあまり利用されない傾向にある．結局，利用者は，パターン検索のような簡単な入力方法を好む．

また，現代人はマルチメディア化により多量の音声情報を与えられているので，実際に綴りを知らずとも音声情報とその意味を知っていることがよくある．

したがって，入力文字列に対して音声の似通った文字を自動的に出力してくれる検索ツールがあればより使いやすい．

3 電子英和辞書検索ツール

本研究では，特に日本語を母国語とする利用者が英単語を検索する時に対象を絞り，正しくない入力に対して正しい語の候補を出力する辞書検索ツールを開発する．

3.1 曖昧検索

本研究では，音声的な曖昧検索をするために入力文字列を soundex への変換を用いる．この変換は次のような特徴がある．

入力文字列に対して，その先頭以外に位置する一部の子音情報だけが取り出され，更にその子音は発音されるときに似通った文字と同じコードが残される．この soundex を介して，辞書と比較することで曖昧検索とする．入力からコード，辞書との比較までのプロセスの概念を図 1 に示す．

soundex は以下の手順で生成される．

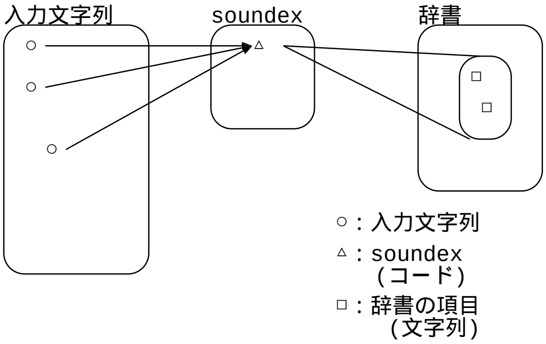

図 1: 曖昧検索の概念図

手順 1 入力文字列の先頭の文字を保持する．先頭以外に位置するすべての A, E, H, I, O, U, W, Y を捨てる．

手順 2 先頭の文字を除いて残った文字すべてに対して，表 1 の数字を割り当てる．

手順 3 もし元の文字列で同じ数字を持つ 2 つ以上の文字が隣り合っていたならば，その先頭以外を除いてすべてを除去する．

手順 4 数字が 3 つよりも少なかった場合，後ろに零を追加するか，3 つより多い場合は，後の数字を捨てることによって，“文字，数字，数字，数字” の形式に変換する．

例えば，単語 “Euler” を soundex へ変換した例を図 2 に示す．

以降，表 1 において数字に変換される文字を子音，変換されず捨てられてしまう文字を母音と呼ぶ．

表 1: soundex へのコード化対応表

文字	変換先の数字
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m, n	5
r	6

3.2 母音情報による順位付け

1 つの soundex に対して一致する辞書の単語は通常複数ある．このため完全一致検索などと違って，soundex を用いた曖昧検索を行うと候補が複数になる．特に soundex では入力文字列の情報を大幅に削除するので，候補数が莫大になると考えられる．そのため順位付けを考えなければならない．

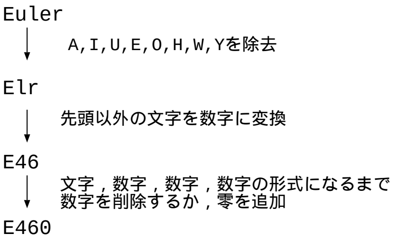

図 2: 単語 “Euler” の soundex への変換例

順位付けでは soundex への変換の際にコード化された子音部分の情報ではなく，まだ利用していない母音部分を用いることにする．図 3 のように子音部分の soundex それぞれに対して決まる 4 つの母音の文字列 V_1，V_2，V_3，V_4 を母音 soundex と呼ぶ．単語を引いた時に入力が誤っていた場合，これらの情報を個人の癖情報として，保存，蓄積することで，順位付けに利用する．誤りの尺度を式 (1)，(2) とする．

$$\sum_{i=1}^{4} err(V_i^{in}, V_i^{out}) \tag{1}$$

$$err(V^{in}, V^{out}) = \begin{cases} 0 & if \quad V^{in}[0] = V^{out}[0] \\ 1/pvh(V^{in}[0], V^{out}[0]) & otherwise \end{cases} \tag{2}$$

ここで，添え字 in，out はそれぞれ，入力文字列，正解文字列のことである．添え字 i は，位置情報である (図 3 参照)．$V_i[0]$ は対応する V_i の先頭の文字を意味し，pvh は個人の母音誤りの履歴情報から作られる関数である．$V^{in}[0]$ が取り得る値を列に，$V^{out}[0]$ が取り得る値を行にした表で，入力文字列が辞書の項目と一致しない場合の個人の履歴を保存，蓄積する．この表により pvh の返す値が決まる．

$$C_1 \quad V_1 \quad C_2 \quad V_2 \quad C_3 \quad V_3 \quad C_4 \quad V_4 \quad \cdots$$

図 3: soundex コード化の位置関係

4　実験と考察

実験に使用するデータを取得するために予備実験とを行ったのち，そのデータを元にして，本研究で作成したツールと UNIX のスペルチェッカである ispell との比較を行った．

4.1　予備実験

朗読された文章を聞き取り，その内容をキーボードで入力することを行った．被験者はシステム工学科の学生 1 名であり．文章の題材は，Andrew S. Tanenbaum の “Computer Networks” Prentice-Hall,1981 である．これにより，正しくない文字列と正しい文字列の対応がとれたデータを取得する．

データは，正しくない文字列を拾い上げて正しい文字列を組にしておく．データの総数は，146 組であった．

4.2　比較実験

予備実験により得られたデータの組を入力として，ispell と本辞書ツールの性能比較をする．入力されるデータは正しくない綴りの文字列であり，ツールが正解を出力したかどうかの判断は，正しい文字列を出力したかどうかによる．

まず，ispell と本研究のツールの検索能力を比較するため，データを入力し候補を出力した回数である出力回数，正解を出力した回数の出力回数を比較した．

本研究のツールでは正解回数は多い．しかし，同時に候補数が非常に多くなってしまっている．

表 2: 検索能力の比較

	ispell	開発した辞書ツール
入力単語数	146	146
出力回数	103	146
正解回数	68	105
候補数 (平均)	4.7	54.4
候補数 (分散)	5.1	44.5

次に，順位付けの効果を検証する．結果を，表 3 に示す．順位付けを用いることで，正解の平均がより上の順位に上がっていることがわかる．また，ばらつきが減少して安定したこともわかる．

表 3: 出力順位の比較

	開発した辞書ツール	
順位付け	有	無
正解の順位 (平均)	33.4	8.80
正解の順位 (分散)	33.2	13.34

更に履歴情報を用いた場合について，出力する候補数を限定できるかどうか調べるために，正解と同じ点数の項目の最終順位を比較する．履歴情報を用いると順位自体はそれほど変化がないが，最大数が減少していることがわかる．

表 4: 履歴情報を用いた順位付けの比較

		履歴情報無し	履歴情報あり
順番	平均	8.80	8.31
	分散	13.34	12.42
順番 (最大)	平均	14.6	10.8
	最大	89	68

5　おわりに

音声的な誤りに対して動作する電子英和辞書検索ツールの開発を行った．その機能は現状のスペルチェッカとも適応範囲においては重複する部分が多い．表 3 のとおり，母音による順位付けの有効性が示されている．また，表 4 のとおり，出力する候補を絞る効果もあり十分利用可能なツールであると考えられる．また，ツール全体サイズも Emacs-Lisp のソースファイルで 25KBYTE と少なく，検索時間も 0.5 秒程度であるので，ストレスなく利用できる．

参考文献

[1] D. Knuth，“The Art of Computer Programming, vol. 3: Sorting And Searching”．Addison-Wesley publishing company，pp.391-392(1973)．