アルファベット列の
かな変換についての考察

平成 16 年 2 月 6 日

情報電子工学科 竹野研究室
田村 大輔

目 次

1 はじめに **1**

2 yomi の概要 **1**

2.1 yomi の特徴 1

2.1.1 perl について 2

2.1.2 kakasi について 2

2.2 動作環境 3

2.3 音声データの種類 3

2.4 yomi の使用方法とオプション 5

3 yomi を音声データに割り当てるまでの問題点 **5**

3.1 漢字変換の誤り 7

3.2 アルファベットの読み 8

4 アルファベットのひらがな変換方法の考察 **9**

4.1 ローマ字のアルファベット列のひらがな変換方法 9

4.2 英語のアルファベット列のひらがな変換方法 11

4.3 英語のアルファベット列の変換規則の作成と考察 14

4.4 実験と考察 19

5 まとめ **24**

参考文献 **25**

概要

竹野研究室で開発された Unix 専用の文書読み上げソフトで yomi というものがある。この研究室では、過去に何度かこのソフトの改良や問題点の考察と打開などが行われてきたが、まだこのソフトの改良点や問題点はあると考えられる。yomi では、アルファベット列を綴り一文字ずつ読む方法をとっているが、この読ませ方では文書中のアルファベット列が聞き取りにくく、認識が困難である。
この論文では、アルファベット列をかな文字に変換する何らかの方法を提案し、その変換方法の問題点をとりあげ、また被験者による実験を行ってその変換方法の実用性を考察し、 yomi にその変換方法が取り入れられるかを考えていく。

1　はじめに

音声出力ソフトは現在 MS–Windows や MS–DOS などでは普及されているが、Unix 上ではあまり普及していなく、MS–Windows のように音声出力が OS で統一されていないことがこの要因である。そのためこの研究室で Unix 専用の文書読み上げソフトとして yomi というものが開発された。これは元々は読み合わせ作業を目的に作られたものあり、過去に数々の研究が行われてきた。

yomi は、漢字やカタカナはひらがなに変換してそのかな一文字ずつを音声データに割り当て、アルファベット列は、綴りを一文字ずつ A ~ Z までの音声データに割り当てる方法をとっている。しかし文書中に出てきたときのアルファベット列は、この方法では非常に聞き取りにくく、またそのアルファベット列を認識することが困難である。

今回の研究では、 この yomi の機能で不十分と思われるアルファベット列の読み方の改良を行う。どんなアルファベット列が与えられても綴りの通りに読むのではなく、英単語もしくはローマ字列として読める方法を考えていき、その方法が、現在ある綴りの通りに読む方法よりよいものにするということを目標とする。

2　yomi の概要

2.1　yomi の特徴

yomi の特徴は大まかに、

1. 音声出力の方法は、単にテキストファイルをひらがなの文書に直して、それに対応するひらがなの音声データを単純につないで音声デバイスなどに出すという方法
2. 日本語化された perl を使用していて perl を知っていれば修正や改良が可能
3. 漢字のひらがなへの変換には kakasi を使用
4. 熟語などの読みにはあまり強くなく、記号やアルファベット列などの読みには向いていない
5. 50 音や数字、アルファベットの音に対応した音声データをつなげる、という方法なので、そのようなデータを録音し直せば任意の声で読ませることが可能
6. 日本語のコードは EUC のみに対応
7. 読みの速度の変更が可能

などが挙げられる。これらについていくつか説明していく。

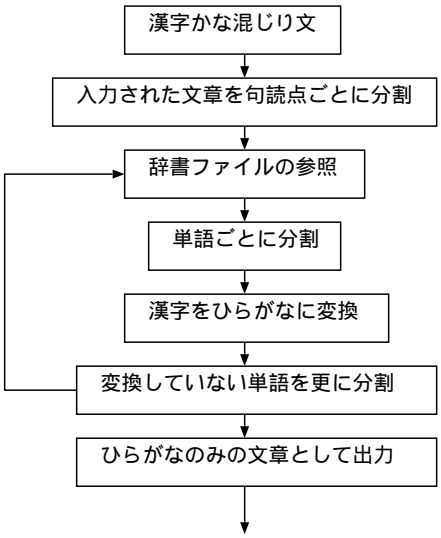

Fig. 2.1 kakasi による処理の流れ

2.1.1 perl について

perl とは Larry Wall 氏が開発した、テキストデータを処理するためのプログラミング言語であり、以下のような特徴がある。

1. Unix や それと同等の機能を持つ OS に向いている
2. テキストの検索や抽出、レポート作成に向いた言語で、表記法は C 言語に似ている
3. インタプリタ型であるため、プログラムを作成したら、コンパイルなどの処理を行なうことなく、すぐに実行することができる

2.1.2 kakasi について

kakasi とは高橋裕信氏が作成した漢字かな変換ソフトであり、このソフトはフリーソフトとしてソースが公開されている。このソフトは漢字、かなが混ざった文章をひらがな、またはローマ字のみの文章に変換し出力するソフトで、漢字の読めない端末を使ったときや、漢字に不慣れな外国人や子供に文章を紹介したい時に使うことを目的として作られた。

yomi に使われている、kakasi の日本語文書をひらがなの文書に直すための処理の流れは Fig. 2.1 のようになっていいる。

また kakasi の特徴としては次のようなものが挙げられる。

1. 文章をひらがな、ローマ字のどちらにも変換できる
2. 熟語の途中に空白や改行が入っていても変換できる
3. 変換後の表示の方法を選べる

2.2 動作環境

基本的な yomi の動作は次の通りである。

1. kakasi で漢字をひらがなに変換して文章を単語ごとに区切る
2. 各単語をひらがな、数字、アルファベット、空白に分類する
3. 分類された単語を音声ファイル名に分割し、コマンドラインを生成する
4. そのコマンドラインを実行する

2.3 音声データの種類

yomi に用いられる音声は 1 音節ごとの音声データとして用いられるものであり、現在使用している音声データの種類は、

1. 50 音データ　([あ]、[か] など)
2. 濁音、半濁音のデータ　([が]、[ぱ] など)
3. 拗音よ含むデータ　([きゃ]、[じょ]、[てぃ] など)
4. アルファベット　([a]、[z] など)
5. 数字 (0 ~ 9 と [ひゃく] や [せん] などの単位)
6. 無音データ

具体的な音声データ表は Table 2.1 、 Table 2.2 通りである。Table 2.1 はひらがなの音声データ、Table 2.2 は数字の音声データである。またアルファベットは、A から Z

読み	file	読み	file	読み	file	読み	file	読み	file
あ	a	か	ka	さ	sa	た	ta	な	na
い	i	き	ki	し	si	ち	chi	に	ni
う	u	く	ku	す	su	つ	tsu	ぬ	nu
え	e	け	ke	せ	se	て	te	ね	ne
お	o	こ	ko	そ	so	と	to	の	no
は	ha	ま	ma	や	ya	ら	ya	わ	wa
ひ	hi	み	mi	ゆ	yu	り	ri	お	wo
ふ	hu	む	mu	よ	yo	る	ru	ん	n
へ	he	め	me			れ	re		
ほ	ho	も	mo			ろ	ro		
が	ga	ざ	za	だ	da	ば	ba	ぱ	pa
ぎ	gi	じ	zi			び	bi	ぴ	pi
ぐ	gu	ず	zu			ぶ	bu	ぷ	pu
げ	ge	ぜ	ze	で	de	べ	be	ぺ	pe
ご	go	ぞ	zo	ど	do	ぼ	bo	ぽ	po
きゃ	kya	しゃ	sha	ちゃ	cha	にゃ	nya	ひゃ	hya
きゅ	kyu	しゅ	shu	ちゅ	chu	にゅ	nyu	ひゅ	hyu
きょ	kyo	しょ	sho	ちょ	cho	にょ	nyo	ひょ	hyo
みゃ	mya	りゃ	rya	ぎゃ	gya	じゃ	ja	びゃ	bya
みゅ	myu	りゅ	ryu	ぎゅ	gyu	じゅ	ju	びゅ	byu
みょ	myo	りょ	ryo	ぎょ	gyo	じょ	jo	びょ	byo
ぴゃ	pya	ふぁ	fa	てぃ	ti	でゅ	dyu		
ぴゅ	pyu	ふぃ	fi	とぅ	tu	っ	ttsu		
ぴょ	pyo	ふぇ	fe	じぇ	je				
		ふぉ	fo	でぃ	dyi				

Table 2.1 ひらがなの音声データ表

読み	file	読み	file	読み	file	読み	file
ぜろ	0	ご	5	じゅう	10	さんびゃく	300
いち	1	ろく	6	ひゃく	100	ろっひゃく	600
に	2	なな	7	せん	1000	はっひゃく	800
さん	3	はち	8			さんぜん	3000
よん	4	きゅう	9			はっせん	8000

Table 2.2 数字の音声データ表

までの 26 ファイルで、空白は null である。実際にはファイル名にはそれぞれ拡張子である “.au” がついている。

なお、音声データを入れ換えれば違った声で読み上げさせることも可能である。

2.4 yomi の使用方法とオプション

yomi を使う上での基本的な形は、読ませたいテキストファイルを

% yomi [テキストファイル]

のようにして実行すればテキストファイルを読み上げてくれる。また複数のテキストファイルを読ませたい場合は、

% yomi [テキストファイル 1] [テキストファイル 2] ....

とすればよい。

またオプションを加えることで、読ませ方や読む速度を変えることができる。オプションの使い方は、

% yomi [オプション] [テキストファイル]

のように テキストファイル名の前に加えればオプションとして機能してくれる。yomi に存在するオプションは Table 2.3 通りである。

一覧をみると ‘-nd’ と ‘-d’ のように ‘n’ がついているものとそうでないものがある。これは効果のところを見てもわかるが、 ‘n’ をつけることでオプションに反対の機能をつけることになるものであり、また ’n’ がついているものには、(default) と書かれているが、これは標準、すなわち、何も指定しない場合の機能である。

3 yomi を音声データに割り当てるまでの問題点

yomi は基本的に低機能であり、また音声の品質があまり良いものではなく聞き取りにくいという問題もある。しかしここでは、与えられた文章をひらがなに変換するまでの部分に関する問題を考え、そしてその問題点についての改善点などを考察する。

オプションの一覧	
オプション	効果
-nd	音声出力を実行 (default)
-d	音声出力はせず、実行コマンドを script の形で 出力
-nt	文の区切りは改行毎 (default)
-t	文の区切りは句点毎
-ni	(数字) の形のものは数字ではなく数の列して読む (default)
-i	(数字) もカッコ、数字、カッコとして読む
-ns	ポーズ (キー入力待ち) なし (default)
-s	改行毎にポーズ (キー入力待ち) を入れる
-nq	画面に現在の行を表示する (default)
-q	画面表示はなし
-nr	画面表示は元々の現在行を使用 (default)
-r	画面表示は kakasi で変換した行を使用
-nn	数字用の音素を使用 (default)
-n	数字もひらがなに直してベタ読みする
-na	アルファベット用の音素を使用 (default)
-a	アルファベットもひらがなに直してベタ読みする
-ao [文字列]	auconn に渡すオプションを指定
-ko [文字列]	kakasi に渡すオプションを指定
-ap [文字列]	音声ファイル再生プログラム部分を指定
-sh	-d オプション時に shless 用のコメントも出力
-v	バージョン の表示
-h	オプション一覧参照

Table 2.3 yomi のオプションの種類

3.1 漢字変換の誤り

これは漢字のひらがな変換に使っている kakasi の問題なのだが、一応ここに上げる。

例 :

私は新潟工科大学生です。

これを yomi で読ませると、

わたしはにいがたこうかだいがくなまです

と、大学生 [だいがくせい] を大学 [だいがく] と生 [なま] の 2 つに分けて変換して、おかしな読み方をしてしまう。

kakasi はこのように変換の際にいくつか誤った変換をする場合がある。この傾向は、複合語に多くみられ、 いくつか例を挙げて説明する。複合語とは 2 つ以上の単語を組み合わせたもので、例えば、[株式会社] (株式 + 会社) のようなものである。

1. 単語を組み合わせることによって読みが濁音になるもの

   例:

   [機械仕掛け] × [きかいしかけ] → ○ [きかいじかけ]

2. 送りがなのある単語で、漢字と漢字の間にひらがながくるものとの組み合わせのもの

   例:

   [喋り方] × [しゃべりほう] → ○ [しゃべりかた]

3. 多くの単語の組み合わせ

   例:

   [新潟工科大学生] × [にいがたこうかだいがくなま] →
   ○ [にいがたこうかだいがくせい]

4. 送りがなの種類がいくつかある単語との組み合わせのもの

   例:

   [時計仕掛け] × [とけいじかけけ] → ○ [とけいじかけ]

kakasi はひらがなに変換する際に、辞書に登録されている一番長い単語から短い単語へと順番に検索していく形をとっている。 1. の場合、 kakasi の辞書には [機械仕掛け] という単語が載っておらず、[機械] と [仕掛け] しかなく別々に変換するので、1. のような誤りが起こってしまう。 2. の場合、kakasi は漢字と漢字の間にひらがながあるとそのひらがなの後ろで単語を区切ってしまい、そのため [喋り方] は、[喋り] と [方] に分けて検索されてしまう。[方] は単一では [ほう] と変換されるため 2. のような誤りが起こってしまう。 3. の場合、最初に述べたように kakasi は長い単語から検索していくようになっていて、[工科大学] という単語が辞書の中にあるため [新潟] と [工科大] と [学生] には分割されず、[新潟] と [工科大学] と [生] に分けられてしまう。[生] は、[なま] と変換されるため 3. のような誤りが起きる。 4. の場合、 1. の例の単語と形は似ているが、これはきちんと連濁ができている。しかし送りがながおかしくなっている。これは、kakasi の辞書には、[時計仕掛](とけいじかけ) と載っていて [時計仕掛け](とけいじかけ) とは載っておらず、[け] を余計に増やしてしまうため 4. のような誤りが起こる。

これらの問題を改善するには、kakasi の辞書に載っている単語を増やせば良い。kakasi はソースが公開されているフリーソフトだからこういったことは可能であるように思える。しかし、この方法でこれら全てを改善しようとすると、辞書の登録単語数が膨大になり、それにより単語の検索に時間がかかりすぎ、あまりよいものとはいえない。他にもソースを改良し、送りがなや連濁を辞書にたよらずに行うという方法があるが、それは簡単ではないように思える。

3.2 アルファベットの読み

yomi ではアルファベットはその 1 文字の読み、例えば ‘a’ なら [エイ] としか読むことができない。よって次の文章

これは yomi で書きました。

は、

これは [ワイオーエムアイ] でかきました

と読んでしまう。

人間が [ワイオーエムアイ] を聞いてそれを ‘yomi’ とわかるまでにはいくらか時間が掛かってしまう。‘yomi’ を [よみ] と読ませるには、アルファベット列をローマ字とみなして、yomi にローマ字をひらがなに変換するプログラムを組み込めば可能であり、それほどは難しい方法ではないと思われる。また、ローマ字ではない英単語のようなアルファ

ベット列が与えられたときも、何らかの変換方法でひらがなに変換すれば、読ませることが可能だと考えられる。

4　アルファベットのひらがな変換方法の考察

§3 で与えられた文章をひらがなに変換するまでの yomi の問題点を挙げた。漢字変換の誤りを改善するのは難しそうであるが、アルファベット列を現在の yomi の方法以外の形で読ませるのは可能だと考えられる。ここでは、 yomi でアルファベット列を、今までと違った方法でどう読ませるかを考える。

yomi の音声出力方法は、文章をひらがなに直して、そのひらがなに対応する音声データをつなげて音声デバイスに出すという方法であり、アルファベット列をローマ字のように読ませる場合も、それを何らかの方法でひらがなに変換さえできれば、現在の yomi にもすぐ活用できると考えられる。

今回はアルファベットのひらがな変換に、テキストファイルをスクリプトとして記述された編集方法に従って一括処理する sed という Unix 標準のテキスト処理コマンドを使用した。

4.1　ローマ字のアルファベット列のひらがな変換方法

ローマ字には日本でよく使われるものとしてヘボン式と日本式がある。ヘボン式と日本式のローマ字表記を Table 4.1 にまとめてみた。

ヘボン式と日本式を比べると、アルファベットの綴りに多少の違いはあるが、同じ綴りで違う読み方をする、といったことはないので混合して使っても特に問題はない。よってアルファベット列をローマ字としてひらがなに変換するとき、ヘボン式と日本式の 2 つのいずれもサポートする変換方法をとればよい。

しかしこの方法でひらがなに変換するとき、

1. 促音 [っ] の表記方法
2. 長音 [ー] の表記方法
3. ‘n’ が含まれるときの判断の仕方

といった問題がでてくる。

a	ka	sa	ta	na	ha	ma
i	ki	shi[si]	chi[ti]	ni	hi	mi
u	ku	su	tsu[tu]	nu	fu[hu]	mu
e	ke	se	te	ne	he	me
o	ko	so	to	no	ho	mo
ya	ra	wa	n	ga	za	da
	ri			gi	ji[zi]	ji[di]
yu	ru			gu	zu	du
	re			ge	ze	de
yo	ro	wo		go	zo	do
ba	pa	kya	sha[sya]	cha[tya]	nya	hya
bi	pi					
bu	pu	kyu	shu[syu]	chu[tyu]	nyu	hyu
be	pe					
bo	po	kyo	sho[syo]	cho[tyo]	nyo	hyo
mya	rya	gya	ja[zya]	[dya]	bya	pya
myu	ryu	gyu	ju[zyu]	[dyu]	byu	pyu
myo	ryo	gyo	jo[zyo]	[dyo]	byo	pyo

Table 4.1 ヘボン式ローマ字と日本式ローマ字の一覧 ([ ] の中が日本式ローマ字)

1. は、日本ローマ字の [っ] の表現の仕方は ‘ tta ’[った] のように最初のアルファベットの子音字を重ねるものである。この場合単に子音字が重なったときに [っ] に変換すればよい。

2. は普通ローマ字の [ー] は母音字の上に ‘^’ や ‘–’ を付けるものだが、コンピュータ上などで書かれたものには、‘^’ や ‘–’ が付いてることが少ないと思われる。例えば、[東京] は、ひらがなをローマ字に直せば、‘ Toukyou ’ だが、[う] を長音と判断することが多くその場合は ‘ Tôkyô’ と書くが、実際 [東京] は、 ‘Tokyo’ で通じるので ‘^’ や ‘–’ を付けることはほとんどない。

3. は ‘n’ の後ろに母音や ‘y’ がついた場合の問題である。例えば ‘ne’ というものがアルファベット列の中にあったとして考える。この場合の読み方は、[ね] と [んえ] の 2 通りである。ローマ字において ‘n’ と ‘e’ を切り離して考えたい場合、厳密には ’(クオート) をいれるものだが、実際にはそういったことはあまり見かけない。よって、この場合は誤変換の可能性がある。

しかし ‘nne’ がアルファベット列の中にあった場合その読み方は、[んね]、[んんえ]、[っね]、[っんえ] の 4 通りが考えられが、 2 番目の [ん] という文字が 2 つ重なる言葉と、 3 番目や 4 番目の [っ] の後にナ行や [ん] がくる言葉はほとんどない。よって、‘nne’ というものがアルファベット列にあったら [んね]([ん] とナ行) と変換すればよい。

このようにいくつか問題はあるものの、ローマ字をひらがなに変換することはそんなに難しくなく大きな問題も起こらない。

4.2 英語のアルファベット列のひらがな変換方法

日本語文章中に出てくるアルファベット列は、そのほとんどが英単語で、ローマ字だけでは読むことはできない。よって、前にあげたアルファベット列をローマ字としてひらがなに変換する方法はほとんど使えない。

英語は子音の数も母音の数も日本語より多いので、英単語をひらがなに直すことは厳密には難しい。しかし外来語ように、英単語を日本語のように読ませるようひらがなに変換することはできるのではないかと思われる。ここでは、その変換方法や変換結果の考察、実際に読み上げてみた場合の認識の度合いに関する実験結果などを述べる。

英単語をコンピュータで読み上げる一番簡単な方法は、単語 1 つ 1 つの読み方の辞書を作る方法で、例えば ‘water’ というつづりの単語に [うぁーた] という発音を登録するといったことを行えば変換自体はそんなに難しいものではない。しかし、その方法ではそのための辞書が必要となり、その作製にはかなり手間がかかる。よってここでは、少ない変換規則で英単語をひらがなに変換する方法について考察する。

computer	Microsoft	excel	word	power	point	information
technology	file	folder	help	mail	internet	command
access	home	page	server	screen	saver	system
Unix	Windows	section	Japan	America	new	save
color	display	camera	version			

Table 4.2 日本人がよく使う英単語

ca[か]	la[ら]	va[ば]	fa[ふぁ]	qa[か]	
ci[き]	li[り]	vi[び]	fi[ふぃ]	qi[き]	wi[うい]
cu[く]	lu[る]	vu[ぶ]	fu[ふ]	qu[く]	wu[う]
ce[け]	le[れ]	ve[べ]	fe[ふぇ]	qe[け]	we[うえ]
co[こ]	lo[ろ]	vo[ぼ]	fo[ふぉ]	qo[こ]	wo[うお]

Table 4.3 ローマ字にない [子音字 + 母音字]

できるだけ少ない変換規則で英単語をひらがなに変換するといっても、すべての英単語に対応するためには、さすがにローマ字の変換表のようなものだけでは足りず、かなり膨大な規則やそのための調査が必要であると考えられる。よってまずは日本人がよく使うような英単語 (外来語や固有名詞) についていくつか挙げて考えてみる。

4.2 に日本語文章中に含まれそうなアルファベット列を適当に 32 個挙げてみた。

この中でローマ字のひらがなの変換規則だけで変換できるものは 'home''page' 'Japan' しかなく、しかも 'home' は [ほめ] 'page' は [ぱげ] と全く意味の通らない言葉となってしまう。しかし日本語のローマ字の母音字と子音字の関係は、英単語にもよくみられるのでいくらか修正や追加をすれば使えると思われる。

ローマ字の構成は子音字 (母音字以外のアルファベット)+母音字 (a,i,u,e,o)、もしくは、母音字や 'n' のみであり、ローマ字のひらがな変換規則で、ひらがなに変換できないアルファベットは、ローマ字にない日本語の子音字と母音字の関係 (ce,fi,le,ve,qu など) か子音字の後ろに母音字がこない場合である。よってこれらをひらがなに変換する規則が必要となる。これを英語の発音記号などを参考にして、Table 4.3 と Table 4.4 に作ってみた。これとローマ字のひらがな変換規則を基に Table 4.2 の英単語をひらがなに変換したものを Table 4.5 に挙げる。

この結果をみると、一応すべての英単語をひらがなに変換できているが、まるで意味の通らない言葉になってしまっているものが多くみられる。これは、[子音字 + 母音字] のアルファベット列のひらがな変換規則を辞書などで調べてもっとも適しているものを探しだし修正すれば、多少は意味の通る言葉が増えてくるが、これだけではほんの少しの英単

b[ぶ]	c[く]	d[ど]	f[ふ]	g[ぐ]	h[]	j[じ]
k[く]	l[る]	m[む]	p[ぷ]	q[く]	r[ー]	s[す]
t[と]	v[ぶ]	w[う]	x[くす]	y[い]	z[ず]	

Table 4.4 アルファベット 1 字 (子音字) のひらがな変換規則 (‘h’ は無音とする)

computer[こむぷてー]	Microsoft[みくろそふと]	excel[えくすける]
word[うおーど]	power[ぽうぇー]	point[ぽいんと]
information[いんふぉーまちおん]	technology[てくのろぐい]	file[ふぃれ]
folder[ふぉるでー]	help[へるぷ]	mail[まいる]
internet[いんてーねと]	command[こむまんど]	access[あくけすす]
home[ほめ]	page[ぱげ]	server[せーべー]
screen[すくれえん]	saver[さべー]	system[すいすてむ]
Unix[うにくす]	Windows[ういんどうす]	section[せくちおん]
Japan[じゃぱん]	America[あめりか]	new[ねう]
save[さべ]	color[ころー]	display[ぢすぷらい]
camera[かめら]	version[べーしおん]	

Table 4.5 作成したひらがな変換規則を用いて Table4.2 をひらがなに変換した結果

語しかまともに変換できないことに変わりはない。この原因について考えられることを次に挙げる。

1. ローマ字と違ってひとつの [子音字 + 母音字] の関係にも読み方はひとつでなく複数存在する
2. ある決まったアルファベットの並びがきたら変換規則が変わることがある
3. ほとんど読み方が一定であるアルファベットの並びがある

1. は例えば ‘ti’ という部分文字列の読み方は日本語に直すとしても、‘tiara’(ティアラ) のように [てぃ] と読むものや ‘time’(タイム) のように [たい] と読むものや ‘ticket’(チケット) のように [ち] と読むものなど複数個でてきてどれがよいかとは一概にはいえないということであり、2. は例えば、 ‘command’ という英単語は、日本語の読み方では [コマンド] で重なっている ‘m’ を 1 つ発音しなくということであり、3. は例えば語尾の ‘tion’ はほとんど [しょん] としか読まなく、アルファベット一字と [子音字 + 母音字] のアルファベット列のひらがなの変換規則では、うまく変換できないということである。

これらの問題をなくすためには、アルファベット一字と [子音字 + 母音字] のアルファベット列のひらがなの変換規則で変換する前に、問題の箇所を例外として処理すればよい

と考えられる。

1. を例外処理するためには、まず例の ‘ti’ のように複数の読み方があるときどのようにしてそれらの読み方が使われるかについて調べる必要がある。ひとつの [子音字 + 母音字] の関係にも複数の読み方があるといってきたが、それは後にくる母音字にの読み方で大きく 2 つにわけられると考えらる。ひとつはローマ字にあるような [a(あ),i(い),u(う),e(え),o(お)] という読み方をするときで、もうひとつはアルファベットの読み方である [a(えい),i(あい),u(ゆう),e(いい),o(おう)] と読むときであり、例の ‘ti’ をみても 3 つの読み方の [てぃ] と [ち] は ‘i’ を [い] と発音するもの、[たい] は [あい] と発音したものと分けることができる。よって [子音字 + 母音字] のアルファベット列のひらがなの変換規則は、母音字をローマ字にあるような [a(あ),i(い),u(う),e(え),o(お)] を基準にするものとアルファベットの読み方である [a(えい),i(あい),u(ゆう),e(いい),o(おう)] を基準にするものと 2 つ作り、アルファベット列の並びなどからそれを場合分けして例外処理させればよいと考えられる。しかしこれを場合分けできるケースはあまり多いとはいえないので、場合分けできないときは、 2 つの母音字の発音を比べて比較的多く使われるローマ字にあるような [a(あ),i(い),u(う),e(え),o(お)] を基準にするもので変換する形をとる。

2. は、変換規則が変わる形がきたらその変わった変換規則に対処できる処理方法を作ればよいと考えられ、例の場合は同じ子音字が重なったら発音するものはひとつになっているから、子音字が重なったら 2 字を 1 字にするという処理を行えばよい。

3. はその並びの変換規則を作って、最初に変換すればよいと考えられ、例の場合は ‘tion’ を [しょん] にする変換規則を作り、これが語尾にあったら最初に [しょん] と変換すればよい。

これらの例外処理をいくつか探し、そしてこの変換方法に組み込んで [子音字 + 母音字] のアルファベット列のひらがな変換規則を修正すれば、多くの英単語をよりわかりやすいひらがなに変換できるのではないかと考えられる。

しかしこの例外処理をいくつか作っても、どんな英単語も完璧に変換するといったことはかなり難しいが、 yomi で読ませるとき、その変換方法で変換したひらがなが多少間違っていても聞き手にその英単語が正確に伝われば問題はないので、アルファベット列を完璧にひらがなに変換する必要はなく、多くの英単語を相手に伝りやすく変換するということを目標にして、アルファベット 1 字 (子音字) のひらがな変換規則や [子音字 + 母音字] のアルファベット列や例外処理の項目について考えていく。

4.3 英語のアルファベット列の変換規則の作成と考察

アルファベット列の変換方法は、アルファベット 1 字と [子音字 + 母音字] のアルファベット列のひらがなの変換規則に例外処理を組み合わせた計 3 点で行う。

アルファベット 1 字 (子音字) のひらがな変換は、 4.4 の変換規則でほぼ問題ないと考えられるのでそのまま使用する。

[子音字 + 母音字] アルファベット列のひらがなの変換規則は、このままでは問題と考えられるので修正を行う。§4.2 で挙げたように変換規則は二つ作る必要があり、これを英語の発音記号や辞書を参考にして作成して Table 4.6 にまとめた。表の左は母音字をローマ字を基準にした変換規則で、右はアルファベットの読み方を基準にした変換規則であり、1 つしかないものは共通の変換規則とする。なお [x + 母音字] と [y + (i,e)] は、[子音字 + 母音字] として変換するより子音字のみと母音字のみで、それぞれ別々に変換したほうがよいと考えられるのでその変換規則は作らない。

また英単語は、ローマ字にある 'y' を拗音ととることがほとんどみられないので、この変換規則も作らない。

最後に例外処理の規則を作成する。例外処理の中にもいくつか例外が生じて対処できないものがあるが、多くの英単語を相手に伝わりやすく変換することが目標なので、多少の例外は仕方が無いものと考える。例外処理は、

- 'tion' もしくは 'sion' という並びがアルファベット列の語尾にあった場合、'tion' は [しょん] と 'sion' は [じょん] と変換する。また 'tion' の前に 'a' があった場合、その前の [子音字 + 母音字] の関係のアルファベット列を Table 4.6 の右のかっこの変換規則で変換する。

  例:

  condition → condi[しょん]

  information → infor[めい][しょん]

- 'book' 、 'new' 、 'ball' 'view' という並びがアルファベット列の中にあった場合、それぞれ [ぶっく]、[にゅー]、[ぼーる]、[びゅー] と変換する。

  例:

  notebook → note[ぶっく]

  news → [にゅー]s

  baseball → base[ぼーる]

  interview → inter[びゅー]

- 'y' がアルファベット列の中にあり、次に母音字がこない場合、'y' が語尾にあるときは、'y' を 'ir' に変換して、それ以外は 'i' に変換する。

  例:

a[あ][えい]	ba[ば][べい]	ca[か][けい]	da[だ][だい]	fa[ふぁ][ふぇい]
i[い][あい]	bi[び][ばい]	ci[し][さい]	di[でぃ][でい]	fi[ふぃ][ふぁい]
u[う][ゆう]	bu[ぶ][びゅう]	cu[く][きゅう]	du[どぅ][でゅう]	fu[ふ][ふゅう]
e[え][いい]	be[べ][びい]	ce[せ][しい]	de[で][でぃい]	fe[ふぇ][ふぃい]
o[お][おう]	bo[ぼ][ぼう]	co[こ][こう]	do[ど][どう]	fo[ふぉ][ふぉお]
ga[が][げい]	ha[は][へい]	ja[じゃ][じぇい]	ka[か][けい]	la[ら][れい]
gi[ぎ][がい]	hi[ひ][はい]	ji[じ][じゃい]	ki[き][かい]	li[り][らい]
gu[ぐ][ぎゅう]	hu[は][ひゅう]	ju[じゅ][じゅう]	ku[く][きゅう]	lu[る][りゅう]
ge[げ][ぎい]	he[へ][ひい]	je[じぇ][じい]	ke[け][きい]	le[れ][りい]
go[ご][ごう]	ho[ほ][ほう]	jo[じょ][じょう]	ko[こ][こう]	lo[ろ][ろう]
ma[ま][めい]	na[な][ねい]	pa[ぱ][ぺい]	qa[か]	sa[さ][せい]
mi[み][まい]	ni[に][ない]	pi[ぴ][ぱい]	qi[き]	si[し][さい]
mu[ま][みゅう]	nu[ぬ][にゅう]	pu[ぷ][ぴゅう]	qu[く]	su[さ][しゅう]
me[め][みい]	ne[ね][にい]	pe[ぺ][ぴい]	qe[くえ]	se[せ][しい]
mo[も][もう]	no[の][のう]	po[ぽ]	qo[こ]	so[そ][そう]
ta[た][てい]	va[ば][べい]	wa[わ][うぇい]	ra[ら][れい]	ya[や]
ti[てぃ][たい]	vi[び][ばい]	wi[うぃ][わい]	ri[り][らい]	
tu[とぅ][ちゅう]	vu[ば][びゅう]	wu[わ]	ru[る][りゅう]	yu[ゆ]
te[て][てい]	ve[べ][びい]	we[うぇ][うぃい]	re[り][りい]	
to[と][とぅう]	vo[ぼ][ぼう]	wo[うぉ][うぉう]	ro[ろ][ろう]	yo[よ]
za[ざ][ぜい]	sha[しゃ][しぇい]	cha[ちゃ][ちぇい]	tha[ざ][ぜい]	wha[わ][うぇい]
zi[じ][ざい]	shi[し][しゃい]	chi[ち][ちゃい]	thi[じ][ざい]	whi[うぃ][わい]
zu[ず][じゅう]	shu[しゅ][しゃう]	chu[ちゅ][ちゅう]	thu[ず][じゅう]	whu[わ]
ze[ぜ][じい]	she[しぇ][しい]	che[ちぇ][ちい]	the[ぜ][ざ]	whe[うぇ][うぃい]
zo[ぞ][ぞう]	sho[しょ][しょう]	cho[ちょ][ちょう]	tho[ぞ][ぞう]	who[ふう]

Table 4.6 [子音字 + 母音字] アルファベット列と母音字のひらがな変換規則

copy → copir

system → sistem

- 違った母音字が二字重なった場合 (ai au ea のみ)、‘ai’ は ‘a’、‘au’ は ‘o’、‘ea’ は ‘e’ に変換して、その前にある子音字と組み合わせてもしくはそれ単独で Table 4.6 の右のかっこの変換規則で変換する。

  例:

  mail → mal → [めい]l

  auto → oto → [おう]to

  cream → crem → c[りい]m

- アルファベットが二字重なった場合、子音字は、2 字を 1 字にして、母音字は 2 字を 1 字にして、さらにその前にある子音字と組み合わせて、Table 4.6 の右のかっこの変換規則で変換する。ただし文字列の頭に ‘acc’ がきた場合にのみこの処理は行わない。

  例:

  office → ofice

  seed → sed → [しい]d

  access → access

- ‘cir’、‘com’ という並びがアルファベット列のの中にあった場合、その後に母音字がこないときそれぞれ [さー]、[こん] と変換する。

  例:

  circle → [さー]cle

  computer → [こん]puter

- ‘er’ という並びがアルファベット列の中にあった場合、‘er’ のあとに母音字がこないとき、 [‘a’ ー] に変換する。

  例:

  poster → posta[ー]

- ‘sh’ 、 ‘th’ 、 ‘ch’ という並びがアルファベット列の中に場合、後に母音字がきていなかったらそれぞれ [しゅ]、[す]、[ち] に変換して、母音字がきていたら、 Table 4.6 のように変換する。また ‘ch’ が語尾にあって前に ‘t’ があった場合は ‘tch’ を [っち] に変換する。例:

wish → wi[しゅ]

three → [す]ree

beach → bea[ち]

watch → wa[っち]

- ‘gh’ という並びがアルファベット列の中にあった場合、‘gh’ は取り除く。

  例:

  eight → eit

- ‘ph’ という並びがアルファベット列の中にあった場合、‘ph’ は ‘f’ に変換する。

  例:

  symphony → symfony

- ‘ck’ という並びがアルファベット列の中にあった場合、‘ck’ が語尾にあったら [っく] に、それ以外は ‘k’ に変換する。

  例:

  back → ba[っく]

  packet → paket

- 語尾に [子音字 + e] がきた場合、その ‘e’ を取り除いて、語尾の [子音字 + e] に一番近い [子音字 + 母音字] の関係のアルファベット列もしくはそれがなければ母音字のみを Table 4.6 の右のかっこの変換規則で変換する。ただし語尾の [子音字 + e] に一番近い母音字の前に母音字がきていた場合は語尾の ‘e’ を取り除くだけとする。またその ‘e’ の前の子音字が ‘c’ と ‘g’ だった場合は、それぞれ ‘ce’ を [す] ‘ge’ を [じ] に変換してからその処理を行う。

  例:

  scale → s[けい]l

  arcade → ar[けい]d

  ice → [あい][す]

  route → rout

  page → pa[じ] → [ぺい][じ]

- アルファベット列が 1 つの [母音字 + 子音字]、もしくは母音字のみの場合、そのアルファベット列は、 Table 4.6 の右のかっこの変換規則で変換する。

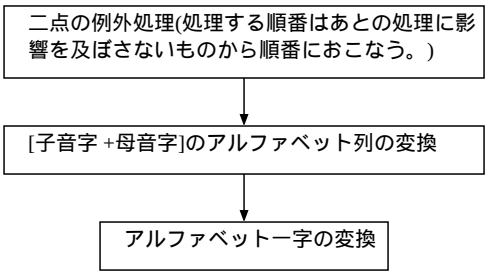

Fig. 4.1 アルファベット列をひらがなに変換するまでの手順

$$\text{he} \rightarrow [\text{ひい}]$$
$$\text{I} \rightarrow [\text{あい}]$$

というものでで処理するものとして、その処理の流れは上の方から順番に行う。

英単語をひらがなに変換するまでの手順は、 Fig. 4.1 のようであり、例として 'section' という英単語をひらがなに変換するまでの流れは次のようになる。

まず 'section' が語尾の 'tion' を [しょん] に変換するという例外処理の項目に当てはまるから、

$$\text{section} \rightarrow \text{sec}[\text{しょん}]$$

となり、他に例外処理の項目に当てはまるものがないので、次に [子音字 + 母音字] のアルファベット列の変換の処理へいき、ここで当てはまるものは 'se' でだけでありこれを

$$\text{sec}[\text{しょん}] \rightarrow [\text{せ}]\text{c}[\text{しょん}]$$

と変換して、最後にアルファベット一字の変換の処理へいき

$$[\text{せ}]\text{c}[\text{しょん}] \rightarrow \text{せくしょん}$$

と変換する。

4.4 実験と考察

新たに作成した変換方法で英単語をどの程度正確にひらがなに変換できるか、それがどの程度人に伝わるかを実験して考察する。

computer[こんぷたー]	Microsoft[みくろそふと]	excel[えくすせる]
word[うぉーど]	power[ぱわー]	point[ぽいんと]
information[いんふおーめいしょん]	technology[てちのろぎー]	file[ふぁいる]
folder[ふぉるだー]	help[へるぷ]	mail[めいる]
internet[いんたーねと]	command[こまんど]	access[あくせす]
home[ほうむ]	page[ぺいじ]	server[さーばー]
screen[すくりいん]	saver[さばー]	system[しすてむ]
Unix[うにくす]	Windows[ういんどうす]	section[せくしょん]
Japan[じゃぱん]	America[あめりか]	new[にゅう]
save[せいぶ]	color[ころー]	display[でぃすぷれいー]
camera[かめら]	version[ばーじょん]	

Table 4.7 新たな変換規則を用いて、 Table4.2 をひらがなに変換した結果

もう一度 Table 4.2 のアルファベット列を変換して、結果を Table 4.7 にまとめる。

Table 4.7 は外来語であり、正確に外来語に変換されているかどうかを判断基準として前の変換規則で変換した Table 4.5 と比べてみる。Table 4.5 の正解数は、 32 個中 4 個であり、Table 4.7 の正解数は、 32 個中 11 個である。正解数は増えたが正解率はまだ低い状態である。しかし 'home[ほうむ]' や 'page[ぺいじ]' といった長音をひらがなにしたものや、 'display[でぃすぷれいー]' や 'information[いんふおーめいしょん]' といった僅かな誤りしかないものなどが、前の変換規則と比べて格段に多くなっていて、そういったものをいれると 7 割近くが正確またはそれに近いものに変換されている。しかしまだ color[ころー] のようにまるで意味がわからない単語に変換してしまうものもあり、まだまだ改良が必要と考えられる。

30 個程度ではどれくらい正確に変換できるか判断しにくいので、今度は外来語のアルファベット列を 200 個ひらがなに変換してみて、外来語に『正確に変換できたもの』、『ほとんど正確に変換できたもの (誤りが 1 字)』、『ほとんど正確に変換できなかったもの』の 3 種類にわける。選んだ外来語はすべてカタカナを辞書から適当に抜きだたもので、その英単語の綴りをひらがなに変換して 3 種類にわけた。結果は次の通りである。

正確に変換できたもの → 200 個中 53 個

ほとんど正確に変換できたもの → 200 個中 64 個

ほとんど正確に変換できなかったもの → 200 個中 83 個

1 字以下の誤りしかなかったものは全体の約 6 割であり、この結果からこの変換方法では、半分以上はほぼ正確に変換できると推測できる。しかしうまく変換できても人が理解

正確に変換できたもの								
変換した単語	その外来語	変換後の単語	正解数		評価			
			カタカナ	英語	1 人目	2 人目	3 人目	4 人目
eight	エイト	えいと	4	4	A	A	A	B
king	キング	きんぐ	4	4	A	A	A	A
star	スター	すたー	4	4	A	A	A	A
test	テスト	てすと	4	3	A	A	A	A
file	ファイル	ふぁいる	4	3	A	A	A	A
drive	ドライブ	どらいぶ	4	4	A	A	A	A
dance	ダンス	だんす	4	4	A	A	A	C
piano	ピアノ	ぴあの	4	4	A	A	A	A
level	レベル	れべる	4	3	A	A	A	A
news	ニュース	にゅーす	4	4	A	A	A	A

Table 4.8 実験結果 1(かっこ内の数字は 2. の結果で変化したもの)

できなければ意味がないので、この変換方法でどのくらい人にその英単語が伝わるか、また実際 yomi に取り入れて使うときは、変換方法を聞き手が知っている状態で使うので、変換方法を知っている場合と知らない場合では認識に変化があるか、などを実験する。

被験者は 4 人で実験方法は次の通りである。

1. まず英単語をひらがなに変換した 200 個の外来語の中から外来語を『正確に変換できたもの』と『ほとんど正確に変換できたもの (誤りが 1 字)』と『ほとんど正確に変換できなかったもの』をそれぞれ 10 個ずつ計 30 個をランダムに並べて何人かの被験者にみてもらい、元の英単語が何かわかるかについてカタカナとアルファベットの両方で答えてもらい、そのわかり方の度合いを [簡単にわかった] → A 、[比較的わかりやすかった] → B 、[どちらともいえない] → C 、[よくわからなかった] → D 、[まったくわからなかった] → E の 5 段階で評価してもらう。
2. 次に変換規則にローマ字の変換規則に似たものを使用していることを伝え、さらに例外処理を行わなずにひらがなに変換するものを例として 1 つみせて、(ここでみせるものは digest(ダイジェスト) → [でぃげすと]) 新たにわかったところや修正したいところをその箇所だけ答えてもらう。
3. 最後に元の英単語とその外来語をみせてそれらを、正確に知っていたか、知らなかったか、間違えて覚えていたかの 3 通りで答えてもらう。

実験 1.、2. の結果は、Table 4.8、4.9、4.10 と次の項目にまとめる。

ほとんど正確に変換できたもの								
変換した単語	その外来語	変換後の単語	正解数		評価			
			カタカナ	英語	1 人目	2 人目	3 人目	4 人目
album	アルバム	あるぶむ	4	4	A	A	A	A
game	ゲーム	げいむ	4	4	A	A	A	A
circus	サーカス	さーくす	1	0	E	C	D	E
extra	エキストラ	えくすとら	0	3	E	A	B	A
tire	タイヤ	たいー	0	0	E	E	E	E
space	スペース	すぺいす	4	4	A	A	B	A
page	ページ	ぺいじ	4	4	A	A	B	A
knife	ナイフ	くないふ	4	2	D	B	D	E
mail	メール	めいる	4	4	A	A	B	A
boat	ボート	ぼあと	3(4)	3(4)	B	B	B	E

Table 4.9 実験結果 2(かっこ内の数字は 2. の結果で変化したもの)

ほとんど正確に変換できなかったもの								
変換した単語	その外来語	変換後の単語	正解数		評価			
			カタカナ	英語	1 人目	2 人目	3 人目	4 人目
comma	カンマ	こま	1	1	D	C	E	E
studio	スタジオ	すとぅでぃお	3	3	B	B	D	C
office	オフィス	おふぁいす	3(4)	3(4)	D	E	B	A
credit	クレジット	くれでぃと	2	1	B	A	D	E
tour	ツアー	とうー	0	0	C	B	C	E
door	ドア	どうー	0	0	D	A	C	A
foot	フート	ふぁうと	0	0	D	E	E	D
dial	ダイアル	でぃある	0	0	E	C	C	E
ticket	チケット	てぃけと	3	3	B	A	D	E
pool	プール	ぽうる	0	0	D	D	D	D

Table 4.10 実験結果 3(かっこ内の数字は 2. の結果で変化したもの)

	カタカナの正解数	英語の正解数
正確に変換できたもの	10 個中 10 個	10 個中 9.25 個
ほとんど正確に変換できたもの	10 個中 7.25 個	10 個中 7.25 個
ほとんど正確に変換できなかったもの	10 個中 3.5 個	10 個中 3.5 個

Table 4.11 正解数の分布

- 実験 2. はほとんど成果がみられなかった。
- 実験 3. で知っていたカタカナの数の平均は 30 個中 29.5 個で、知っていた英語の綴りの数の平均は 30 個中 23.75 個であった。
- カタカナの正解数の平均は 30 個中 20.75 個 、英語の正解数の平均は 30 個中 20 個で、正解数の分布は Table 4.11 のようになっている。
- 正確なカタカナを知っていて答えられなかった数の平均は 8.75 個で英語の綴りを知っていて答えられなかった数の平均は 3.75 個である。
- 正確なカタカナを知らなくて答えられたもの、及び正確な英語の綴りを知らなくて答えられたものはなかった。

実験 1. では、変換に誤差がでるほど正解率も悪くなっていき、1 字の誤りを含むものでもひらがなの文字数が少ないものや、全く違った読み方のひらがなに変換しているものでは正解率が悪くなっている。しかし 2 字以上の誤りを含むものでも、ひらがなの文字数が多いものや、似たような読み方のひらがなに変換しているものでは正解率は高くなっている。またわかりやすさの評価も変換に誤差がでるほど悪くなり、ばらつきがでてくる。

実験 2. でほとんど成果がなかったのは、 1. でわからなかったものがアルファベット 1 字や、例外処理で誤りを起こしているものが多いからと考えられる。しかし英語の綴りを知らなくて答えられたものがないことや、カタカナの正解数を英語の正解数が上回っていることがほとんどないことから、被験者はひらがなから英語の綴りを導き出しているのではなく、ひらがなを自分の知っている単語に当てはめて、その単語の英語の綴りを自身の知識から出して答えていると考えられる。よって変換方法を知っていても、人がその変換方法から単語を導き出す可能性は低く、英単語の綴りを想像することは難しく、 yomi に実装する場合も、変換方法を知っていてもあまり効果がないと考えられる。

この結果と前の 200 個の外来語のアルファベット列をひらがなに変換した結果から、今回の方法では適当な 200 個の外来語のアルファベット列をひらがなに変換した単語をみて元の単語が何かわかるものは、カタカナで 200 個中 128.5 個 、英語で 200 個中 124.5 個、すなわち、外来語の英単語を変換したものならおよそ 6 割が相手にみて伝わるといってもよいのではないかと思われる。

しかし、英単語の綴りに関しては、被験者がひらがなを自分の知っている単語に当てはめてその単語の英語の綴りを自身の知識から出して答えていると考えられることから、綴りを知らない英単語を、綴りを含めてこの方法で相手に伝えることは難しいと思われる。また最終的には音声にするということを考えると今よりさらに人が認識する可能性が低くなると推測できるので、変換規則に関してもまだ改良が必要と考えられる。

5 まとめ

今回の研究は、 yomi のアルファベット列の綴りをそのまま読む方法以外の方法を提案し、その方法による変換結果とそれに関する考察を行い、それを数人の被験者にためしてもらい、問題点と実用性の考察を行った。今回考えた方法は、アルファベットに一字と [子音字 + 母音字] の変換規則を作り、それに例外処理を追加することでアルファベット列をひらがなに変換させるもので、簡単なプロトタイプを用いて実験を行ったが今後もさらに研究が必要であると思われる。

今回の方法で、アルファベット列をひらがなに変換した結果や実験をみると、現在の例外処理ではおよそ 6 割くらいを人が認識できると推測されるが、例外規則を今より多く見つけて増やしていけば、もっと高い割合で人が認識できると考えられ、この方法でもよい変換ができるのではないかと思われる。しかし実験では、英単語を対象とした単語のみで、しかも音声ではなく文字で行っていて、

- ローマ字と英単語の区別はどうするか
- 音声でどこまで伝わるか
- 文章中に出てきた場合それがアルファベット列を変換したものとわかるか

などの問題がまだ数多くあり、また知らない英単語を伝えることはこの方法では原理的に難しいと考えられ、今回のような方法を yomi に取り入れて、現在ある変換方法よりよいものになるかどうかは今のところ何ともいえなく、これらは今後の課題であろう。

参考文献

[1] 新村出編 「広辞苑」第五版 岩波書店, pp2962-2963

[2] 佐藤健美 :“UNIX における日本語文書読み上げソフトの開発について”, 新潟工科大学卒業論文, (1999)

[3] 井上直大 :“日本語文書中の漢字説明ソフトの開発について”, 新潟工科大学卒業論文, (2000)

[4] ホームページ : IT 用語辞典 e-words, http://e-words.jp/

[5] 研究所 「新英和大辞典」第五版 , ppx4-x6