3階層モデル準拠の新しいコーパス解析システムにおける英語スペルチェッカーの改良

3階層モデル準拠の新しいコーパス解析システムにおける英語スペルチェッカーの改良
(An Adaptation of English Spellchecker in a New Corpus Analysis System Based on the Three-Tier Model)*

岡田　毅

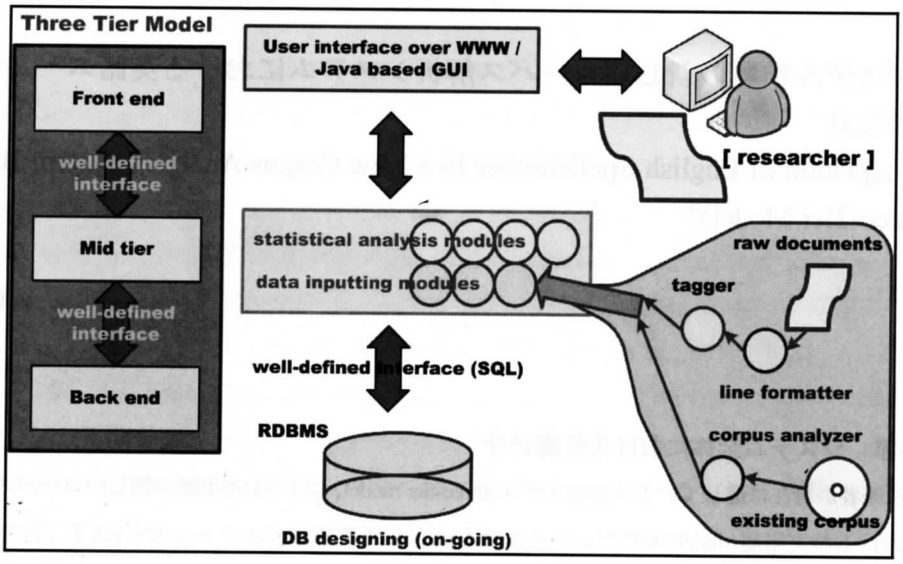
1. 序節：システム全体における位置付け

本論は筆者が現在推進している3階層モデル(three-tier model、以下3TM)準拠の新しいコーパス解析システムにおける特定階層(tier)中のモジュールのひとつとして英語スペルチェッカー(以下 SPCHK)を位置付け、コーパスの基幹データとなる一次入力(primary input)ソースの作成および入手方法に応じた柔軟な処理を実現するプログラムとしてのSPCHKの重要性とその性能の向上に向けた試みを論じるものである。

3TMはPetrou, et als (1999)でも論じられているように、Web上での一種の分散型商用処理システムで広く用いられているモデルであり、情報工学の分野では一定の評価を得ているものの、コーパス言語学の処理システムにはこれまで応用されることがなかった。その結果、大規模なコーパスや世界規模でのコーパス共同研究のプロジェクトが既に始動し、その必要性が強く認識されているとはいえ、コーパス言語学の分野における研究では、依然として個人研究者ないし小規模研究グループによる独自開発のコーパスおよびそれに呼応した独自開発の処理システムによる解析処理がその主流を占めている。

Okada and Sakamoto (2009)では、リレーショナルデータベースマネジメントシステム(以下RDBMS)を基盤とするコーパス処理システムに対してこの 3TM の概念を応用することによって得られるシステムの効率化と世界規模での共同研究の発展の可能性について言及した。図1は3TMの概念(左端)と実際のコーパス処理システムの対応を示したものである。3TMになぞらえると、コーパス解析システムにおいて、エンドユーザーに最も近い位置でWebやJava準拠のGUI(graphic user interface)を実現する層がFront end (tier)、統計処理を中心とする各種のコーパス解析を担う層がMid tier、全ての解析の対象となるコーパスそのものがSQLにより蓄えられ管理される層がBack end (tier)に相当する。Mid tierは統計処理モジュールと、柔軟な属性付与のなされた多量のデータをRDBMSへと受け渡す機能を果たすデータインプットモジュールに大別される。このデータインプットモジュールの中の、line formatter(行整形プログラム)の一部として、処理対象となる英文の性格に応じたフィルタの1つとしてSPCHKを位置付けることになる。[注1]

図 1.　Three Tier Model とコーパス解析システム

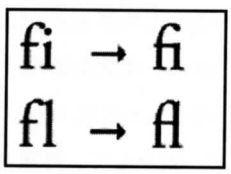


既存コーパスに対し、一定の解析を施し RDBMS で処理しやすい統一的な形式に整えるのもこのモジュールの重要な機能のひとつである。これと並行してコーパス化されていない「生の」英文データの形式を整え、必要な属性を付与していくという機能があってこそ、ユーザー自身が必要とするデータを汎用的な形式でデータベース化することが保証される。この「生の」データは様々な手法で作成され、また多様な文書特徴を持っているといえる。英文をキーボードから手入力することによって作成された文書には、人間の指の動きに起因するエラー以外に、英文を作成した人間の母語の種類別による独特のエラーも含まれることになる。ここで、本論で扱うような日本人学習者の犯すエラーに特化して調整された SPCHK が威力を発揮することになる。

それについて論じる前に、紙媒体の英文をコンピュータ可読な形式に変換する最も一般的な手法である光学的文字認識(OCR)処理の過程で生じるエラーの特徴とその対処方法について概観してみよう。OCR を経た結果に特徴的に現れるのは、合わせ活字(ligature)と、単語や文字の間の空白(スペース)の誤認識の問題がある。

合わせ活字とは、ラテン語やその後の中世の手書きスクリプトにおいて、書く速度を上げるために 2 つ以上の文字を繋げて書き表したことから始まり、15 世紀に活字式の印刷技術が発明されてからも、活字の中には多くの合わせ活字が含まれていた。これは活字そのものの数を減じることにも効果的であったが、20 世紀初頭にかけてタイプライターが誕生してからは、アルファベット式の言語の場合、それぞれの文字を別個の活字でタイプすることになった。しかし、1970

図 2.　合わせ活字



年代以降、コンピュータ写植が普及するにつれて、図2のAdobe Caslon Proに見られるような合わせフォントが復活することになり、あたかも手書き文書時代のような文字の連鎖が多く見られるようになった。これらの合わせフォントは、コンピュータ内部ではそれぞれ固有の文字コードを与えられているが、一旦それらがプリンタを介して紙の上に印字された場合、それらをコーパスの一次資料としてプレーンな文字コード化してコンピュータに取り込もうとする際には、OCR処理を施す以外の方法がなく、その過程において独特の問題が発生することになる。すなわち、画像の上では単一のフォントとして印字されている本来は2文字の組み合わせを、2つの文字として改めて分解して認識し文字コードを与え直さなければならないのである。

2つの隣接する文字の間にふさわしい空白が介在すればOCRはそれらを別個の文字として認識するが、コンピュータ写植の進歩によって文字間や単語間に介在する空白の幅は一定ではなくなっている。OCRの設定の際に、空白幅を極端に狭く指定してしまえば、単語中の文字と文字の間に発生しているわずかな幅の空白を単語同士を区切る空白であると誤認識することがあるし(例: *beat* を *be_at*)、反対に空白の幅を広く定義してしまえば、単語間の空白が認識されずに複数の単語が繋がって認識されてしまうことになる(例: *be_ at* を *beat*)。誤認識の結果が明らかな非英単語である場合、Microsoft Word 搭載の市販SPCHKでも正しい分割を促すこともあるが、これは必ずしも全ての場合にあてはまるわけではない。例えば(1)のような誤認識部分に対して、WordのSPCHKは対応できない。

(1) Like most enlightenedzoologicalgardens, Cotswold Wildlife Park is a member of The Federation of Zoological Gardens of Great Britain and Ireland. (Cotswold Wildlife Park (1992) BNC(AM2))

しかし、表1に示すような、外国語・記号・略号等を除外したLOB (Lancaster-Oslo/Bergen) Corpus中の54038種類(延べ1008050単語)中の2文字連鎖データから得られた「存在しない2文字連鎖」情報に基づくと、(2)の網掛け隣接部分が、単語間の空白をOCRが認識し損なった結果生じた連鎖であると正しく判断できる。

(2) enlightenedzoologicalgardens

そして、融合してしまった複数単語の分離用のフィルタプログラムが機能し、正しい単語連鎖である(3)を発生させる。

(3) enlightened zoological gardens

表 1.　LOB コーパスには存在しなかった 2 文字連鎖

aa	cp	fp	hg	jj	kq	pc	qj	sv	vf	wm	xs	zf
aq	cv	fq	hj	jk	kv	pd	qk	sx	vg	wp	xv	zg
bc	cw	fv	hk	jl	kx	pf	ql	sz	vh	wq	xw	zh
bf	cx	fw	hp	jm	kz	pg	qm	td	vj	wt	xx	zj
bg	cz	fx	hq	jn	lb	pj	qn	tj	vk	wv	xz	zk
bh	db	fz	hv	jp	lg	pn	qo	tk	vl	ww	yd	zm
bk	dc	gb	hw	jq	lj	pq	qp	tq	vm	wx	yg	zn
bm	df	gc	hx	jr	lq	pv	qq	tv	vn	wy	yh	zp
bq	dq	gd	hz	js	lx	pw	qr	tx	vp	wz	yj	zq
bv	dx	gf	ih	jt	lz	px	qs	tz	vq	xb	yk	zr
bw	dz	gj	ii	jv	md	pz	qt	uh	vr	xd	yn	zs
bx	fb	gk	ij	jw	mj	qa	qv	uj	vs	xf	yq	zt
bz	fc	gq	iw	jx	mk	qb	qw	uk	vt	xg	yu	zu
cb	fd	gv	iy	jy	mq	qc	qx	uq	vv	xj	yv	zv
cd	fg	gw	jb	jz	mv	qd	qy	uu	vw	xk	yx	zw
cf	fh	gx	jc	kc	mw	qe	qz	uv	vx	xl	yy	zx
cg	fj	gz	jd	kg	mx	qf	rj	uw	vz	xm	yz	
cj	fk	hb	jf	kj	mz	qg	rz	uy	wc	xn	zb	
cm	fm	hc	jg	kk	oj	qh	sj	vb	wg	xq	zc	
cn	fn	hf	jh	kp	pb	qi	sr	vd	wj	xr	zd	

先に分離するか先に SPCHK をかけるか、という処理順序の問題をめぐって、単一のプログラムで処理を実現するのではなく、プログラムを複数のモジュール化されたフィルタ群から構成される複合体としてそれを設計することにより、対象となる文書の作成方法に対応したフィルタの選択や適応順序の変更によって、より精度の高い一次入力データを作成しようとするのが根本的な発想である。これにより、対象となる文書が異なる度に、処理系プログラム全体を書き換える、ないし入れ替えるという非効率性からの脱却が保証され、他のフィルタ群に影響を与えることなく、データ構造に依存することもなく特定対象の特性に呼応したフィルタセットの選択が可能となるのである。

詳述する紙面の余裕はないが、本論で触れている line formatter は一種のトークン化プログラム(tokenizer)であり、単語中の文字連鎖を正しく処理することに加えて、特に文末のピリオドの識別もその大きな役割のひとつである。[注2] 文境界を正しく認識することは、文末の句読点や記号を正しく識別することと表裏一体であり、とりわけ *U.S.A.*などの略字(abbreviation)に付随するピリオドと文末の完全終止符のピリオドの区別が深刻な課題として存在し続けている(これについての議論は Schmid (2008)、Mikheev (2002)、Ratnaparkhi (1996)などを参照)。しかし、文脈情報や略字リストのような外部辞書を活用しても、tokenizer による完璧な文末認定は不可能であり(Shmid(2008))、判断を人間に委ねなければならない場合もある。そのような事例に対しては柔軟で使いやすいグラフィックユーザーインターフェイス(GUI)を 3TM における Front tier に実装することによって実質的には効率の高いデータ取り込みが可能となる。

2. エラーコーパスと SPCHK

過去 10 年の間に、学習者コーパス(ある特定の言語 (多くの場合は英語)を数年間に渡り学んできた非母語話者から集められたテキストの集積)に対する関心は高まりをみせた(Granger,1998a)。これらのコーパスにみられる非標準的なスペリングは、文法や語彙といった高次の言語特徴に対する研究にも繋がった。しかし、この種の研究では分析結果を故意に歪めることを防ぐために、データ入力過程で実際と異なるエラーが生じることを最小限に留めることに注意が払われる必要がある(Granger, 1998b)。例として、Granger and Wynne (2000) は学習者コーパスにおいて高い頻度で生じるさまざまなミススペリング(以下 MSP)が、type/token ratio (総語数に対する異語数の比率)といった語彙的濃度分析の信頼性を損ねてしまう危険性を示している。彼らの研究では、オランダ語、フランス語、ポーランド語、スペイン語母語話者の英語コーパスからデータが採取されたが、彼らは興味深いことに“a mere glimpse at the respective lists shows that each national group has its own specific problems,”(各言語データから得られたリストを瞥見するだけで、それぞれの母語話者が抱える問題が分かる)とし、これらのリストが“prove useful to adapt tools such as spellcheckers to the needs of non-native users” (SPCHK のようなツールを非英語母語話者ユーザーのニーズに適応させるために有益であるとの証明になる)であろうと論じている。

Mitton(1996) はこれより数年前に同様のことを示唆している。彼は英国英語向けの SPCHK を開発する過程で、もし英語学習者が各自の母語干渉による誤りを犯すのならば、ドイツ語話者向けの SPCHK、スペイン語話者向けの SPCHK というように、ユーザーの母語に起因する MSP に対処可能なように SPCHK を改良することが可能であると提案した。Okada (2005) では日本語母語話者(Japanese native speakers、以下 JPNS)による MSP の大規模コーパスを作成し、その分析を行った。これまでにも Brown(1970) や Ziahossiny and Oller (1970)のように、母語話者と非母語話者の MSP の差異に関する調査は試みられてきたが、日本語を母語とする英語学習者の MSP は、コーパスに集約されることも体系的に分析されることもなかった(詳細は Okada (2005)を参照)。

Okada(2005)の研究が Mitton の提案を本論で紹介する実際の SPCHK 改良へと結びつけたことになる。[注3] 本論の目的は、(1)JPNS の英語 MSP 特徴には何らかのパタンがみられるのかを分析し、(2)もし、ある種のパタンが存在するならば、それらに対応できるよう SPCHK を改良できるのかを探り、(3)改良された SPCHK の性能はどの程度向上するのかを、序節で示した 3TM 準拠のコーパス解析システム全体の枠組みの中で検討することである。

3. JPNS 産出の MSP の特徴

本研究においては、2 種類の英語 MSP コーパスが使用された。ひとつは Mitton により 1980 年代に作成されたもので、Oxford Text Archive から数年間提供されていたものである。このコーパス中の NS(native speakers、以下 NS)産出データが利用された。もう一方はさらに近年、Okada により構築された JPNS 産

出による MSP コーパスである。[注4]

2 つのコーパスは、効率良く比較を行うことを目的に厳密な統制の元で対応させたサンプルを収集したものではない。各コーパスは異なる方法を用い、異なる時期に異なるタイプの書き手から成るグループから集められたデータを元としたサブコーパスの集積になっている。しかし、これらの比較が有益であることは以降の議論を通して明らかになる。もしも、データの限られた箇所からは得られないことが明らかな MSP パタンが抽出できるのならば、ここでの比較調査が正しい分析結果を導出していることの証拠となる。

当然のことではあるが、初級の書き手の MSP はエラーコーパス全体に対して有意義な貢献をなさないことが多い。JPNS には中学生が含まれており、彼らの英語力は極めて限られているし、その一方で NS のコーパスには 9 歳の子どもたちや読み書き能力の不十分な成人のデータを含んでいる。そのため彼らの中には、単語を綴ることに深刻な困難を抱えた者もおり、彼らの産出したいくつかの MSP は、結果的に目標単語から大きくかけ離れたものとなっているからである。

JPNS による MSP の大部分は、NS によるそれとほとんど相違がないということを最初に指摘するべきであろう。例として、*disappoint* の MSP を比較してみよう。左側のコラムが NS のコーパスから、右側が JPNS のコーパスからの MSP である。

表 2 では JPNS、NS それぞれの特性が結果の正規分布を歪めたことが明らかである。つまり、多くの書き手に共通する 1 種類の MSP と、それより生起頻度の低い 3、4 種類を除いては、一人の書き手によってのみ産出された数多くの MSP が、結果として MSP の分布パタンを歪にさせているのである。表 2 からは *disappoint* という語のスペリングに関して、具体的には、*s* が 1 つであるのに対し *p* を重ねること、そして(*s*、*p* に比べて割合は少ないが)母音字の使い方に関して、JPNS と NS が、共通の困難点を抱えていたことは明白である。177 個の JPNS による *disappoint* のスペリングのうち、88 パーセントに当たる 156 個は NS 産出の MSP リストにも認められた。しかし、このような背景にもかかわらず、コーパスからは下に掲げる 3 タイプの MSP が JPNS 独自のものとして顕著であることが判明する。

1)　*r* と *l* の置き換え(substitution)、または *l* と *r* の置き換え(substitution)
2)　*v* と *b* の置き換え、または *b* と *v* の置き換え
3)　特に *o* や *u* を伴った余計な母音字の挿入(insertion)

これらのエラーを JPNS が犯す傾向にあることはよく知られており、決して驚くべきことではない。日本語の話し言葉では音素/l/と/r/、/b/と/v/は区別されず、日本語をアルファベットで表記するローマ字のシステムにおいては、*l*/*r* 双方の音に *r* という文字を、*b*/*v* の音には *b* を用い、文字として *l* と *v* は全く用いられない。従って、JPNS にとっては、英語の *r* と *l* (または *b* と *v*)を区別することが NS よりもはるかに困難なのである(Okada (2005)を参照)。同様に、日本語の音声は「子音＋母音」の音節で大部分が構

表 2. *disappoint* の MSP

NS 産出コーパス	頻度	JPNS 産出コーパス	頻度
disapoint	100	disapoint	96
dissapoint	81	dissapoint	42
dissappoint	32	dissappoint	8
dessapoint	29	desapoint	5
disopoint	5	dicapoint	5
dissapiont	5	dispoint	3
dissipoint	3	diccapaint	1
diserpoint	2	dicepoint	1
dissopoint	2	disapaint	1
deapoit	1	disapint	1
deinpent	1	disapoind	1
derepoint	1	disapointment	1
desapoint	1	disapoit	1
desappoint	1	disappint	1
desippoint	1	disappo	1
dessipoint	1	disepoint	1
disapint	1	diserpoint	1
disapiont	1	disipoint	1
disapont	1	dissaopint	1
disappiont	1	dissaspoint	1
disappointment	1	disserpoint	1
disepoit	1	disspoint	1
dispoint	1	dusapoint	1
dispont	1	thispoint	1
dispot	1		
disserpoint	1		
dissippoint	1		
dissopite	1		
dissypoint	1		

成されているという特徴があるために、例えば英語の *br*、*dr*、*tr* といった子音連鎖に、JPNS は母音字を挿入する傾向がある。*library* に対する JPNS 産出の MSP はこのような傾向を鮮明に示しており、JPNS 産出のコーパスには *libelary*、*liberary*、*liburally*, *liburary*, *liveraly*, *liverary*, *liverely* などのような MSP が含まれている。

表 3 では、まず *l, r, b, v* の使われるべき回数をカウントしている。例えば、JPNS 産出のコーパスには *library* に対して 40 種類の MSP がみられたが、もし、この語が正しく綴られていた場合には、40 個の *l* と 80 個の *r* が用いられるはずである。我々は MSP の中で、*r* が用いられるべき箇所に *l* が、反対に *l* が用いられるべき箇所に *r* が現れている回数を計測した。従って、例えば *libelary* のような MSP の場合には、*l* が *r* に代わって用いられた回数を 1 回ずつ付加して計算した。最終的に、*r* と *l* の置き換え、および *b* と *v* との置き換えの割合を計算した。表の( )内の数字は回数を示している。

表3.　*r* と *l*、*b* と *v* の置き換え

	NS 産出コーパス	JPNS 産出コーパス
r に対する *l* の誤用	0.5%　(16126)	7.0%　(2708)
l に対する *r* の誤用	0.8%　(13532)	16.5%　(1698)
b に対する *v* の誤用	0.7%　(2906)	10.0%　(982)
v に対する *b* の誤用	0.9%　(2350)	18.1%　(171)

JPNS が *o* と *u* という母音字を挿入して余分な音節を付加してしまうという傾向は表4で明らかである。*covenant* と *pavement*、*naked* と *raked* を比較しても分かるように、一般的に、ある単語に含まれる音節数をスペリングのみから推測することが常に可能とは限らない。従って、MSP に現われた音節数を計測する方法が推測の域を出ない場合も有り得るが、大抵は理論的な予測をすることが可能である。次節で扱う SPCHK には、MSP 中の音節数を推定するモジュールが実装されているため(Mitton(1996)で詳述)、ここでの分析にその機能を用いることとした。

表4. 目標単語と MSP 中の音節数比較

	NS 産出コーパス	JPNS 産出コーパス
母音の o や u の挿入により MSP が目標語よりも多くの音節を持つ	1.2%	3.9%
ほかの理由により MSP が目標語よりも多くの音節を持つ	8.3%	7.7%
MSP が目標語と同数の音節を持つ	68.0%	75.1%
MSP が目標語よりも少ない音節を持つ	22.5%	13.3%
総計	(33740)	(4962)

4.　SPCHK の改良

多くの SPCHK が、上で整理した JPNS に特徴的な MSP に対しては十分に機能しない。例えば Microsoft Word 付属の SPCHK は、前述した *library* に対する JPNS 産出の7種類の MSP のうち、*liberary* と *liburary* のわずか2例に対してしか *library* を正しい修正候補として提示することができない。特に語頭近くで発生した *l* と *r* または *b* と *v* の置き換え MSP は(Okada(2005)参照)、結果として SPCHK プログラムに内部辞書の誤った個所、つまり誤った文字から始まる単語群への検索を促してしまう可能性が高い。

序節で言及した SPCHK は、特定パタンを伴った MSP に対処するために、かなり簡単に改良することができる。しかしその改良方法に触れる前に、この SPCHK が通常モードでどのように機能するのかを概説する必要があるだろう。この SPCHK は、基本的に Wagner(1974)や Veronis (1988)の最小編集距離(minimum-edit distance)の考え方に基づいている。

この手法の核となるのは、目標単語と MSP を、左から右へ向かって 1 文字毎に比較していくというアルゴリズムである。この比較の過程では、MSP の文字(letter of the misspelling、以下 LM)と目標単語の文字((letter of the target、以下 LT)は以下の3つの観点で比較される。(1)その LM は誤って挿入されているものか(insertion)、(2)LT が省略されていないか(omission)、(3)その LM は LT と置き換えられたもので

はないか(substitution)。

例えば、SPCHKが*fown*というMSPの目標単語候補として*phone*を取り上げる場合を考えてみよう(同様に*flown*、*town*など他の多くの単語が目標単語となる可能性もある)。目標単語*phone*をどう処理すればMSPである*fown*を導き出すことができるのだろうか。アルゴリズムは以下のように判断を下していく。(1)*p*が*f*に置き換えられ、(2)*h*が省略され、(3) 次の*o*は正しく、(4)*w*が挿入され、(5)*n*は正しく、(6)*e*が省略されている。このようにして、SPCHKはMSPの*fown*から目標単語*phone*を得るためには、少なくとも4つの編集(edit)操作(1つの置き換え、1つの挿入、2つの省略)が要求され、両者間の編集距離は4であると結論付けるのである。

しかしながら、このような簡単な計測手法はいささか単純すぎる。*fown*の目標単語が(*phone*ではなく)*stone*であると仮定してみよう。すると*fown*と*stone*の間でも、さきほどと同様に編集距離は4になることがわかる。しかし、*fown*を産出した書き手は*stone*ではなく確かに*phone*と綴ろうとしたのではないだろうか。従って*fown*と*phone*の方が*fown*と*stone*の関係よりもりも密接だということを示すアルゴリズムが必要になる。

この種の音声に関する情報をSPCHK内蔵の辞書に組み込むことで、この課題を解決することができる。*phone*に対して、その*ph*部分には*f*という文字が置き換えエラーの可能性となるという情報を与えることにより計算アルゴリズムは、この置き換えとそれに伴う*h*の省略には距離を減じた計算をするという正しい扱いをできるようになる。同様に*phone*の*one*部分に対して*own*がエラー候補としてあり得るという情報を与えておくこともでき(*own, grown, blown*というような類似した音を持つ単語が存在するからである)、その結果、*phone*と*fown*との間の編集距離は短縮されるのである。これに対して*stone*に関する音声的な情報からは*st*の部分に対して*f*をエラー候補として許容するいかなる関連性も導出できない。従って、*s*に対する*f*の置き換えと、*t*の省略はそれぞれ通常の編集距離1ずつとして計算される(*phone*の場合と同様に*one*を*own*で置き換えてしまったエラーは音声情報に基づいて距離を減じて計算される)。このようにして、*fown*というMSPは*stone*ではなく*phone*の方に距離的に近いMSPであると判断されるのである。[注5]

このようなアルゴリズムに従って、SPCHKはMSPを発見すると、内蔵辞書の中から目標単語候補として数百語を検索し、その各々をMSPと比較し、それぞれがどの程度うまく適合するのかを評価するのである。そして候補語毎に編集距離を計算し、距離順にそれらを並べ替える。最もよくマッチしたもの、つまり目標単語との間が最短編集距離のものを最上位に置いて、上位のいくつかをユーザーに提示するのである。

JPNS向けにSPCHKを改良することは比較的たやすい。*phone*の場合に*f*を*ph*の置き換えエラーであると予測させたのと同じように、*library*に対するJPNS産出のMSPの場合でも、*v*を*b*の誤りとして予測させたり、*b*を*v*、また*l*を*r*、*r*を*l*の置き換えエラーというように処理できるようプログラムを手直

しすればよいのである。SPCHK 本体のアルゴリズムや処理系を修正するのではなく、対象とする MSP 群に特徴的なパタンに対処できるようにプログラムの一部のモジュールを差し替えるだけですむのである。

5. SPCHK 改良の効果

MSP に対して、SPCHK は最も妥当性が高いと判断された単語を最上位に置いて、数百にのぼる修正候補つまり目標単語のリストを生成する。目標単語が判明していれば、この候補リスト中にその語が現れるか、またもし現われているのなら最上位にどれほど近い順位を与えられているかを調べることができる。本論で用いている数千種類の MSP を蓄積したコーパスではそれぞれの MSP にその目標単語が対応する形をとっている。従って、実際の英文書ではなく、MSP コーパスに対して SPCHK をかけて、それぞれの MSP に対する修正候補リストを生成させ、そのリストのどの位置に目標単語が現われているかを記録し、分析することができる(目標語が SPCHK の参照する内蔵辞書に含まれていない場合には、それらの MSP をこの調査の対象外とした)。　本論では 2 種類の MSP コーパスと、オリジナルの NS 産出 MSP 用のもの、一方は JPNS 産出の MSP 向けに改良した 2 種類の SPCHK を持っており、表 5 で示すように、4 組の結果が得られることになる。

表 5.　2 種類のコーパスに対する 2 つの SPCHK の性能比較

	NS 産出コーパス	NS 産出コーパス	JPNS 産出コーパス	JPNS 産出コーパス
	NS 用 SPCHK	JPNS 用 SPCHK	NS 用 SPCHK	JPNS 用 SPCHK
1 番目	54.2%	53.3%	61.2%	65.8%
1 - 3 番目	67.9%	67.3%	73.3%	78.7%
6 番目まで	73.4%	73.0%	77.9%	83.5%
総計	35612	35612	4848	4848

表の最初のコラムは、オリジナル版の SPCHK が NS 産出の MSP コーパスにおける MSP に対して、正しい単語を候補単語リストの 1 番目に提示した割合が 54.2%であり、上位 3 番目までにそれを提示した頻度が 67.9%、上位 6 番目までにそれを含んでいた割合が 73.4%であったことを示している。

正しい単語を候補として提示できた成功率が低く感じられたとしても、最下行の「総計」の数字は種類数であって述べ数ではないことに注意すべきである。つまり、各 MSP はコーパス中で 1 度だけ生じたというカウントになっているのである。例えば、正しく *disappoint* を提示できた *disapoint* という MSP は、極めて出現回数が多いエラーであるにもかかわらず、単に 1 回の成功例として計算されている。また正しく目標語を提示できなかった *deapoit* という MSP の場合、これはたった 1 人の書き手によって産出されたエラーだが、1 回の失敗として計算されているのである。

予測の範囲内ではあるが、2 列目に示されているように、NS 産出コーパスに対して走らせた場合、JPNS

向け SPCHK の性能は NS 用の SPCHK のそれよりも若干劣っている。第 3 コラムが示すように、NS 用のオリジナル SPCHK は JPNS 産出コーパスに対してかなりの好成績を収めている。これは既に述べたように、JPNS 産出の MSP の大部分が NS のそれと酷似していることからも予想がつく。しかし、第 4 コラムで示されている数値は重要なものである。それらは、JPNS 産出のコーパスに対して JPNS 向けに改良された SPCHK を稼働させた場合、オリジナルの SPCHK よりも相当高い精度で正しく目標語を提示することができることを明示している。

JPNS 産出の MSP に対する改良 SPCHK の優れた成績は、ほとんど完全に *l* / *r* と *b* / *v* に関わる特徴的なエラーパタンにプログラムを対応させたことに起因している。また SPCHK を JPNS による余分な音節の挿入にも対応できるように改良したわけであるが、この改良が性能にとりわけ大きな差をもたらすことはなかった。それはただ単に、オリジナルの SPCHK が余分な母音に対してかなり寛容である一方、子音に対しては厳密に対処するよう設計されているためである。

先述のように、本研究で用いられた JPNS 産出のコーパスは、NS 産出コーパスと同様に、複数のサブコーパスから構成されており(付録 1 を参照)、MSP データの提供者となった書き手の英語能力にもかなりの差異がみられる。書き手の一部は特に本研究で焦点が当てられた種の MSP をより多く産出する傾向にあった。また、彼らは全く同じ目標単語を与えられて綴ろうとしたわけでもなかった。従って JPNS 向けに改良された SPCHK は、特定のいくつかのサブコーパスでより高い性能を示すことがある。付録 2 にこれらの結果が示されている。程度の差こそ小さいが全てのサブコーパスに対しておいて性能の違いが現われている。付録 3 では、この 2 種類の SPCHK の性能を *albatross* という目標単語に対して JPNS が産出した 111 種類にのぼる MSP に関して比較している。

6. 結論

本研究では、(1)JPNS 産出の MSP には目立った特徴があること、(2)NS 向けに作成されたオリジナルの SPCHK は比較的容易に JPNS 産出の MSP に対応すべく改良することが可能であること、(3)JPNS 産出の MSP に対しては、本研究で紹介した改良が、劇的ではないにせよ意義のある SPCHK の性能向上に繋がること、の 3 点を 3TM に基づいたコーパス解析システムの枠組みの中の Mid tier で line formatter プログラムが果たすべき役割を踏まえながら指摘した。

注

* 本論は学術振興会科学研究費補助金(基盤研究(C) 課題番号: 19530773)の交付を受けて行われている研究の一端を取り上げたものである。

1. これら 2 つのモジュールにとってのパラメータ設定等は GUI などのインターフェイス機能が整備できれば Front end tier での処理となる。
2. トークン(token)とは単語、数字、句読点類、かっこ類、引用符類などの項目を指す。
3. 以降、本論で言及するオリジナルの SPCHK は、Mitton(1996)で紹介されているシステムを指す。

4. 2つのコーパスの詳細については付録1を参照。
5. この問題に関する詳細な論考はMitton (1996)を参照。

参考文献

Brown, D. H. (1970). "Categories of spelling difficulty in speakers of English as a first and second language". *Journal of Verbal Learning and Verbal Behavior,* 9, 232-236.

Furugouri, T. and K. Hiranuma (1987). "Statistical characteristics of English sentences written by the Japanese and detecting and correcting spelling-errors". *Mathematical Linguistics,* 16, 16-26.

Granger, S. (ed.) (1998a). *Learner English on Computer.* London: Longman.

Granger, S. (1998b). "The computer learner corpus: a versatile new source of data for SLA research". In S. Granger (ed.) *Learner English on Computer.* London: Longman, 3-18.

Granger, S. and M. Wynne (2000). "Optimising measures of lexical variation in EFL learner corpora". In J. M. Kirk (ed.) *Corpora Galore.* Amsterdam: Rodopi, 249-258.

Mitton, R. (1996). *English Spelling and the Computer.* London: Longman.

Mitton, R. and T. Okada (2007). "The adaptation of an English spellchecker for Japanese writers". In *Proceedings of the Symposium on Second Language Writing: Second Language Writing in the Pacific Rim,* 6, 15.

Mikheev, A. (2000). "Tagging sentence boundaries", In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics,* 264-271.

Mikheev, A. (2002). "Periods, capitalized words, etc". *Computational Linguistics,* 28, 289-318.

Okada, T. (2003). "Spelling errors made by Japanese EFL writers: a corpus-based approach". paper presented at Colchester Second Language Acquisition Workshop 2003, University of Essex, Colchester, UK.

Okada, T. (2004). "A corpus analysis of spelling errors made by Japanese EFL writers". *Yamagata English Studies,* 9, 17-36.

Okada, T. (2005). "Spelling errors made by Japanese EFL writers: with reference to errors occurring at the word-initial and the word-final position". In V. Cook and Bassetti, B. (ed.) *Second Language Writing Systems.* Clevedon: Multilingual Matters, 164-183.

Okada, T. and Y. Sakamoto (2009: forthcoming). "A new DBMS and flexible POS tagging for EFL learners", In *Proceedings of CL 2009 (Liverpool)*

Petrou, C., S. Hadjiefthymiades and D. Matrakos (1999). "An XML-based, 3-tier scheme for integrating heterogeneous information sources to the WWW", In *Proceedings of Tenth International Workshop on Database and Expert Systems Applications: 1999,* 706-710.

Ratnaparkhi, A. (1996). "A maximum entropy model for part-of-speech tagging", In *Proceedings of the Conference on Empirical Methods in Natural Language Processing,* 133-142.

Reiley, M. D. (1989). "Some applications of tree-based modelling to speech and language indexing", In *Proceedings of the DARPA Speech and Natural Language Workshop,* 339-352.

Schmid, H. (1994). "Probabilistic part-of-speech tagging using decision trees", In *Proceedings of the International Conference on New Methods in Language Processing,* 44-49.

Schmid, H. (2008). "Tokenizing and part-of-speech tagging", In A. Lüdeling and Kytö, M. (ed.) *Corpus Linguistics: An International Handbook (Volume 1).* Berlin: Walter de Gruyter, 527-551.

Veronis, J. (1988). "Computerized correction of phonographic errors". *Computers and the Humanities,* 22, 43-56.

Wagner, R. A. and M. J. Fischer (1974). "The string-to-string correction problem". *Journal of the A C M,* 21,

168-73.
Ziahosseiny, S. M. and J. W. Oller, Jr. (1970). "The contrastive analysis hypothesis and spelling errors". *Language Learning*, 20, 183-189.

付録

付録1: 使用したコーパス

日本人学習者によるスペリングコーパスは5060個の異なったMSPから成る。これは1184個の目標単語に対する延べ数で12000個以上のスペリングの集積である。5060個というのはMSPのトークン(token)数ではなくタイプ(type)数である。例えば、*height*を綴ろうとする試みの中で、*hight*というMSPは延べで181回も出現するが、コーパスの中では1つの項目として記載されている。エラーコーパスは下記の7つのサブコーパスの集合体である。

(1) AEMH-error.txt

244人の日本人大学生(内201名が英語専攻)の手書きエッセイから得られたMSP集。延べ総単語数は20299語で、この内の393単語がMSPであり、重複を除くと234目標単語に対する296種類のMSPが抽出できる。オリジナルソースのURLは以下のとおり。

http://www.lb.u-tokai.ac.jp/lcorpus/data/asao01/

http://www.lb.u-tokai.ac.jp/lcorpus/data/asao02/

http://www.lb.u-tokai.ac.jp/lcorpus/data/shitara01/

(2) EXAMS-error.txt

The Birkbeck Spelling Error Corpus (http://ota.ahds.ac.uk) に含まれるEXAMS.DATのうちの日本人49名の産出したMSPデータ。151個の目標単語に対する162種類のMSP。Cambridge First Certificate in Englishのcomposition試験における作文の中から抽出したMSP集。

(3) HELC-JR-error.txt

日本人中学生(20名から120名と被験者数に幅がある)の和文英訳課題において産出されたMSP集。英訳対象の日本文は286センテンス、総数で85120英単語。431の目標英単語に対する延べ3366個、1921種類のMSP。オリジナルソースは広島大学の三浦省吾監修の*Hiroshima English Learners' Corpus No.1*に収録。

(4)　HELC-SR-error.txt

上と同様のサブコーパス。被験者が 40 名から 120 名の日本人高校生。対象日本文は 68 で産出された英単語総数は 40638 個。187 個の目標語に対する延べ 673 個、346 種類の MSP。オリジナルソースは広島大学の三浦省吾監修の *Hiroshima English Learners' Corpus No.2* に収録。3、4 共に下記を参照。

http://home.hiroshima-u.ac.jp/d052121/eigo1.html

(5)　SAMANTHA-error.txt

「日本語の定義→英単語」形式のスペリングテストから抽出された日本人大学生産出の MSP 集。53 個の設問で被験者数は 333 名。延べ 7418 個、2071 個の MSP。SAMANTHA Error Corpus という名称のオリジナルコーパスは東北大学の岡田毅による。

http://www.intcul.tohoku.ac.jp/okada/corpora/Samantha/Samantha-top.html

(6)　SUZUKI-error.txt

不特定数の日本人高校生の書いた授業中のショートテストにみられた MSP 集。山形県立南陽高等学校の鈴木通明教諭(当時)による。43 個の目標単語に対する 46 種類の MSP。頻度情報は含まれていない。

(7)　FRGRI-error.txt

英語を専攻としない 88 名の日本人大学 1 年生の書いた英作文の中から抽出した、324 個の目標単語に対する 366 種類の MSP。Furugouri and Hiranuma (1987)でリストとして紹介されている。このサブコーパスにも頻度情報は含まれていない。

比較のために、Oxford Text Archive から提供されている Birkbeck Spelling Error Corpus の中の 13 個のファイルを結合することによって、NS によるエラーコーパスが構築された。そのコーパスは 6000 語近い目標単語に対する延べ 220000 個、35000 種類以上にのぼる MSP から成る。表 A1 は各ファイルの一覧である。3 種類のアメリカ英語コーパス中では英国式スペリングとは差異のある目標単語はあらかじめ除外してある。残りの 10 個のファイルは全て英国式スペリングのものである。それぞれのエラーコーパスに関する詳細は Birkbeck Spelling Error Corpus 付属の出典・属性ファイルを参照。本論で用いられた対照エラーコーパスは下記の Web ページで公開中。

http://www.intcul.tohoku.ac.jp/okada/corpora/Atsuo-Henry/

表 A1. NS エラーコーパスの内わけ

ファイル名	出典(ソース)	目標単語数	述べ数	MSP 数
CHES	202 人の 10 歳の子ども	30	2474	1364
FAWTH1	アメリカの印刷物	739		809
FAWTH2	3 人の(スペリングの未熟な)成人	484	1084	557
GATES	New York の小中学生	3390	144179	4401
MASTERS	アメリカの中高生 + 大学生	264	43755	13020
NFER1	83 人の成人書き方(literacy)の生徒	40	838	495
PERIN1	42 人の中学校生徒+成人生徒	61	807	640
PERIN2	6 人の成人書き方の生徒	538	658	625
PERIN3	176 人の 14 歳から 16 歳の生徒	40	1678	901
PETERS1	156 人の 9, 10, 11 歳の子ども	290	18304	10556
PETERS2	925 人の 15 歳の生徒	1618	4147	2576
UPWARD	163 人の 15 歳の生徒	576	1073	753
WING	40 人の大学受験生	185	237	191

付録 2: JPNS 産出の MSP コーパスに対する 2 種類の SPCHK の性能評価

下の表は NS 用の既存 SPCHK と、JPNS 用に改良された SPCHK を JPNS 産出のエラーコーパスに対して稼働させた場合の性能を比較している。表の中での数値は、SPCHK が正しい修正候補単語を候補リストの第 1 位に提示したパーセンテージである。これらは改良された SPCHK の優れた性能を示している。

表 A2. 2 種類の SPCHK の性能比較

処理対象	NS 用 SPCHK	JPNS 用に改良された SPCHK	MSP 種類数
SUZUKI	80.4%	89.1%	46
SAMANTHA	55.2%	63.9%	1953
AEMH	71.2%	74.5%	274
HELC-SR	73.4%	75.5%	327
EXAMS	81.5%	83.4%	151
HELC-JR	59.5%	61.2%	1878
FRGRI	82.9%	83.2%	368

これらの性能スコアに対して t 検定をかけ、また、2 種類の SPCHK 共に正しい修正候補単語が修正候補リストのトップ 10 に掲げられた場合に、その記載順に従って、順位を「スコア」として点数化して付与した。トップ 10 に正しい単語が示されなかった際にはスコアとして 11 が与えられた。当然、t 値はスコアの相違およびサブコーパスに含まれる MSP の数によっても左右される。SAMANTHA, AEMH, HELC-SR, HELC-JR に対して 2 つの SPCHK の性能の差に関しては 0.05 水準で統計的有意差がみられた。MSP 数の少ないせいで SUZUKI に対する性能差では 0.1 水準でしか差異はみられなかった。EXAMS と FRGRI に対しては、統計的有意差はみられないが、他のサブコーパスに類似の傾向は確認される。

付録 3: *albatross* に対する JPNS 産出の MSP

次の表は、2 種類の SPCHK が *albatross* という目標単語に対して JPNS が産出する多くの種類の MSP をどのように処理するかを示したものである。*albatross* は JPNS にとっては綴りにくい目標単語と考えられる。*l* と *b* を含み、*lb* と *tr* の位置では JPNS は母音字を挿入したくなるからである。

第 1 コラムは MSP そのもの、第 2 コラムはそれぞれの MSP を産出した書き手の人数を示している。第 3 と第 4 のコラムは NS 用の SPCHK が提示した修正候補単語についての情報を示している。第 3 コラムは *albatross* という(正しい)目標単語が修正候補単語リストの何番目に現れたのかを示している(0 というのは、この目標単語がリストに出現しなかったことを示す)。第 4 コラムは、それぞれの MSP に対して修正候補リストの第 1 番目に示された単語を表す(従って第 3 コラムが 1 の場合、第 4 コラムは全て *albatross* となる)。第 5、第 6 のコラムは対照的に JPNS 向けに改良された SPCHK の出力を示している。2 つの SPCHK の出力における主要な相違は、JPNS 向けの SPCHK は *ar* で始まる MSP に対して高い性能を示していることである。*ar* で始まるものの中で *arbatross* に対してだけは、NS 用 SPCHK の性能は JPNS 向けのそれと匹敵している。これは目標単語から 1 文字だけ異なる MSP に対しては SPCHK は特別のルーチン処理を行うように設計されているからである。

表 A3. *albatross* に対する 2 種類の SPCHK の性能比較

		NS用SPCHK		JPNS向けSPCHK	
MSP	頻度	リスト中の位置	第1候補	リスト中の位置	第1候補
albatros	56	1	albatross	1	albatross
arbatros	28	0	arbiters	1	albatross
albatoros	27	1	albatross	1	albatross
arbatross	16	1	albatross	1	albatross
albatrous	9	1	albatross	1	albatross
albatlos	7	1	albatross	1	albatross
arbatoros	7	0	arbiters	1	albatross
arbatrous	7	0	abattoirs	1	albatross
arubatros	7	0	abattoirs	1	albatross
alvatros	6	1	albatross	1	albatross
alubatros	5	1	albatross	1	albatross
albatoross	4	1	albatross	1	albatross
albatolos	3	1	albatross	1	albatross
albatorous	3	1	albatross	1	albatross
albutros	3	1	albatross	1	albatross
arbatolos	3	0	abattoirs	1	albatross
arubatoros	3	0	arbiters	1	albatross
arvatros	3	0	avatars	1	albatross
albatloss	2	1	albatross	1	albatross
albatorose	2	1	albatross	1	albatross
albatos	2	1	albatross	1	albatross
albatrose	2	1	albatross	1	albatross
albertros	2	1	albatross	1	albatross
albertross	2	1	albatross	1	albatross
albertrous	2	1	albatross	1	albatross
aldatoros	2	1	albatross	1	albatross
alvatoros	2	3	elevators	1	albatross
alvatross	2	1	albatross	1	albatross
alvatrous	2	1	albatross	1	albatross
arbatloss	2	0	abattoirs	1	albatross
arubatoross	2	0	arbiters	1	albatross
arubatross	2	0	arbiters	1	albatross
ulbatross	2	1	albatross	1	albatross
albadoross	1	1	albatross	1	albatross
albathross	1	1	albatross	1	albatross
albatlas	1	1	albatross	1	albatross
albatlaus	1	1	albatross	1	albatross
albatlus	1	1	albatross	1	albatross
albatorsu	1	1	albatross	1	albatross
albatous	1	1	albatross	1	albatross
albatrce	1	1	albatross	1	albatross
albatris	1	1	albatross	1	albatross
albatroth	1	1	albatross	1	albatross
albatrros	1	1	albatross	1	albatross
albattlos	1	1	albatross	1	albatross
albattros	1	1	albatross	1	albatross
albattrous	1	1	albatross	1	albatross
albertolose	1	3	albatrosses	1	albatross
albertrose	1	1	albatross	1	albatross
albtoros	1	1	albatross	1	albatross
albtros	1	1	albatross	1	albatross
albtross	1	1	albatross	1	albatross
albtrus	1	1	albatross	1	albatross
albutoros	1	1	albatross	1	albatross
albutross	1	1	albatross	1	albatross
albutrous	1	1	albatross	1	albatross
aldatolos	1	1	albatross	1	albatross
allatoroce	1	26	illiteracy	35	illiteracy
albutoloss	1	1	albatross	1	albatross
allubatros	1	1	albatross	1	albatross
alubatolos	1	1	albatross	1	albatross
alubatoros	1	1	albatross	1	albatross
alubatorose	1	1	albatross	1	albatross
alubatoross	1	1	albatross	1	albatross
alubatorous	1	1	albatross	1	albatross
alubatos	1	1	albatross	2	elevators
alubatras	1	1	albatross	1	albatross
alubatrosu	1	1	albatross	1	albatross
alvatoloss	1	5	alveolars	1	albatross
alvatrose	1	1	albatross	1	albatross
alvatrus	1	1	albatross	1	albatross
alvutross	1	1	albatross	1	albatross
arbatlos	1	0	abattoirs	1	albatross
arbatoras	1	0	abattoirs	3	abattoirs
arbatrros	1	0	abattoirs	1	albatross
arbatrus	1	0	arbiters	1	albatross
arbertoros	1	0	arbiters	3	arbiters
arbertros	1	0	arbiters	1	albatross
arbertross	1	0	arbiters	1	albatross
arbertrous	1	0	abattoirs	2	abattoirs
arbtros	1	0	arbiters	1	albatross
arbutros	1	0	arbiters	1	albatross
ardatros	1	0	audacious	1	albatross
arrbatros	1	0	abattoirs	1	albatross
arrubatroce	1	0	aerobatics	3	aerobatics
arubatoras	1	0	arbiters	3	arbiters
arubatoroc	1	0	arbiters	3	arbiters
arubatorous	1	0	arbiters	1	albatross
arubatrous	1	0	abattoirs	1	albatross
arubats	1	0	Arabist	9	Arabist
arubtros	1	0	arbiters	1	albatross
arubutros	1	0	arbiters	1	albatross
arubuts	1	0	abuts	15	abuts
arudatorosu	1	0	audacious	7	audacious
aruvatoros	1	0	aviators	1	albatross
arvatoras	1	0	aviators	6	aviators
ulbatolce	1	8	oblations	7	orbital
ulbatorous	1	1	albatross	1	albatross
ulbatros	1	1	albatross	1	albatross
ulbertorous	1	3	liberators	3	liberators
ulbtlos	1	1	albatross	1	albatross
ulvatrous	1	3	olive-trees	1	albatross