E-014

「誤りやすい英単語を認識させる手法」における似ている規則の距離手法的考察

金子　美和*
Miwa Kaneko

１．はじめに

「誤りやすい英単語を認識させる手法」[1] において誤りやすい英単語だけに母国語を対訳表示する事で，誤使用を発見し易くさせる手法を紹介した．また，その手法の基礎となる「誤りやすい語」の定義とその抽出アルゴリズムを紹介した．この手法において「誤りやすい語」とは，「似ている語」および「非頻出語」であると定義した．「似ている語」の抽出方法として，「似ている規則」を考案し hamming distance の派生形に近い手法を採用し，評価結果を報告した．本稿では，「誤りやすい英単語を認識させる手法」に追加し，「誤りやすい語」の辞書を作成するための文字列の「似ている規則」を 2 章で，採用した手法と従来手法との比較を 3 章で説明し，4 章で採用にいたった理由を説明する．

２．誤りやすい語の辞書と似ている規則

「誤りやすい英単語を認識させる手法」においては，図 1 に示される実行時のアルゴリズムによって，英文をもとに「誤りやすい語」の候補を抽出する．与えられた英文を形態素解析し，未知語(*1)を除いた単語に対して，「誤りやすい語」辞書に定義した単語と照合して対訳付きで抽出する．ここで抽出されなかった単語に対して，ストップワード(*2)でフィルタリングした非頻出のイディオムと単語を英日辞書で対訳を付けて抽出する．ストップワード機能は形態素解析エンジンが備えている機能をそのまま利用する．「誤りやすい語」辞書には，綴りや発音が「似ている語」を対訳付きで登録する．辞書には，ユーザー定義の誤りやすい語も登録可能であるため，似ている語辞書ではなく「誤りやすい語」辞書となっている．誤りやすい語辞書の作成，および形態素解析の英単語辞書への頻出語に対するストップワード属性の追加は，あらかじめ行っておく．

「似ている語」は，形態素解析用の英単語辞書から規則で抽出する．この抽出規則は，"Common Errors in English"[2] の誤りやすい 212 組の分析を行なって定義した．採用された「似ている」規則は，２つの文字列間の「距離」を計算し，文字列長の長いほうを分母とし，類似度を求めるものである．（図 2）この手法は，結果的に，２つの文字列間の「距離」を計算する方法である Hamming distance の派生形となった．

*(*1) 未知語とは，辞書に登録されていない単語．既存のスペルチェッカーはこの未知語に対して正しい綴りの候補を提供する．*

*(*2) ストップワードとは，情報抽出や情報検索などの対象から除外する語で，前置詞，冠詞，be 動詞などの単語．*

String distance based approach for similarity rule in "Method for non-native English users to detect misusage of English words"
* Miwa Kaneko
Software Development Laboratory – Yamato(YSL), IBM

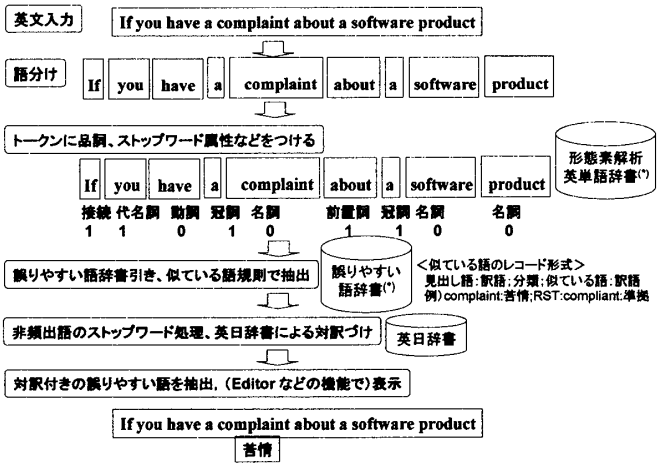

図 1. 実行時アルゴリズムと辞書

(*) 形態素解析の英単語辞書への頻出語に対するストップワード属性の追加，誤りやすい語辞書の作成はあらかじめ行っておく．

距離：$d(x,y)=\sum_{i=0}^{\max(m,n)} a(x_i,y_i),\quad a(x_i,y_i)=\begin{cases}0\ (x_i=y_i)\\ 1\ (otherwise)\end{cases}$

類似度：$p(x,y)=\begin{cases}1-\dfrac{d(x,y)}{\max(m,n)} & (x_0=y_0)\\ 0 & (otherwise)\end{cases}$

$Str(x_1,x_2,x_3....x_n)$　　$Str(x_n,x_{n-1},x_{n-2}....x_1)$

$Str(y_1,y_2,y_3....y_n)$　　$Str(y_n,y_{n-1},y_{n-2}....y_1)$

d を上記2ケースについて求め，それらのmaxをとる

m,n：文字列 x,y の文字列長

図 2. 似ている規則

3. 従来手法との比較

3.1 従来手法

文字列間の距離 (String Distance) とは，２つの文字列間の違いを測る尺度の１つであり，文字の挿入，削除，置換などにより，２つの文字列を等しくするまでの操作の最小回数である．また，この操作の回数のことをコストと呼ぶ．

一般的な文字列間での距離の測定方法には，（1）Edit distance （２）Hamming distance （３）Longest common subsequences (aka LCS) などがある．Edit distance は，最も一般的な手法であり，２つの文字列を等しくする操作として挿入，置換，削除を許可する．Hamming distance は置換のみ許可し，LCS は挿入，削除を許可する[3]．LCS に関しては，文字列間の最長共通部分列（LCS）を求める手法であり，類似性を求めるために使用される．これらは，文字列のあいまい検索，スペルチェック等で，文字列間の違いを判定するアルゴリズムなどに利用されている．例えば，

Edit distance では，alternate と alternative を同じ文字列にするには alternate の最後の e を i に置換し，v と e を挿入する，あるいは，alternative の i を e に置換し，v と e を削除する（コスト 3）．Edit distance の距離は以下のように表現される．[4]

```
d("", "") = 0
d(s, "") = d("",s) = |s|
d(s1+ch1,s2+ch2)
  = min( d(s1, s2) + (if ch1=ch2 then 0 else 1),   ; 置換
    d(s1+ch1, s2) + 1,                             ; 挿入/削除
    d(s1, s2+ch2 + 1)                              ; 削除/挿入
```

また計算量は，edit distance と LCS は O(|s1|*|s2|)，hamming distance は max(|s1|, |s2|) となる．

3.2 文字列間の距離算出方法

"Common Errors in English" では文字列の長さにおいて，以下の分類を行うことができる．これら分類ごとに前述の算出方法を当てはめてみる．例を用いた結果を表 1 に示す．今回は，英単語の類似度を見るという目的のため，文字の位置と順序を考慮し，先頭あるいは最後尾を合わせた Sequential な文字カウントを行った．これは，似ている語を調べた結果，ほとんどの語で同じ位置の文字，あるいは近傍の文字が異なる場合が多いという事実に基づく．

（1） 文字列長が同じ場合
　　（例） adapt/adopt
（2） 文字列長が異なる場合
　（ア）先頭の文字が一致する場合（先頭を合わせ先頭から数える）
　　（例） continual/continuous
　（イ）先頭の文字が一致しないで，最後尾の文字が一致する場合（最後尾を合わせ最後尾から数える）
　　（例） aural/oral
　（ウ）（ア）と（イ）両方の場合（（ア）（イ）でコストが少ないほうを採用する）
　　（例） canon/cannon

表 1: 文字列長ごとの距離

	例	コスト/計算量	Edit distance	Hamming distance	LCS
文字列長が同じ	adapt/adopt	コスト	1（置換1）	1（置換1）	2（挿入1 削除1）
			ch1 or ch2	ch1 or ch2	(ch1 or ch2) x 2
		計算量	25	5	25
文字列長が異なる	先頭一致 continual/continuous	コスト	3（置換2 削除1）	3（置換3）(*1)	5（挿入2 削除3）
			max(ch1, ch2)	max(ch1, ch2)	ch1+ch2
		計算量	90	10	90
	最後尾一致*2 aural/oral	コスト	2（置換1 削除1）	2（置換2）(*1)	3（挿入1 削除2）
			max(ch1, ch2)	max(ch1, ch2)	ch1+ch2
		計算量	20	5	20
	先頭・最後尾一致 canon/cannon	コスト	3（置換2 挿入1）	3（置換3）(*1)	5（挿入2 削除3）
			max(ch1, ch2)	max(ch1, ch2)	ch1+ch2
		計算量	30	6	30

*(*1) 文字列長が異なる場合の hamming distance は，以下のように，不一致部分に仮文字をおくと仮定し，その仮文字と対応する文字の置換と看做す.*

Au|ral
▮o|ral

*(*2) 最後尾一致の場合は，1 番目の文字から一致させて処理を行うと，最長連続文字列をとることができないことが多いので，先頭が一致しない場合は最後尾の文字を合わせ，処理を行う.*

4. 考察

Hamming distance は置換だけを許可するために，同位置文字の距離を測る手法である．似ている語は同じ位置の文字，あるいは近傍の文字が異なる場合が多いということが "Common Errors in English" を調べた結果よりいえるため，Hamming distance は有効な手法である．文字列長が異なる場合は，仮文字を置くと仮定することによって，使用可能である．つまり，先頭一致の場合は，最後尾に，最後尾のみ一致の場合は先頭に，文字列長が同じになるように仮文字を fill in すると考える．今回のカウント手法においては，コスト的な観点からみた場合，edit distance, hamming distance が最小コストとなる．計算量では hamming distance が最小となる．LCS に関しては，共通文字列長（類似度）を求める場合は，適切な手法であると思われるが，今回の手法においてはコスト，計算量ともに高い手法である．以上より，ロジックとして簡潔であり，計算量が少なく，仮文字使用によって拡張性を備えた Hamming distance を採用することになった．この計算方法を用い，"Common Errors in English" を分析した結果，類似度が 0.5 以上の組は 201 組で全体の 94.8% をカバーした．残りの 11 組 (accede/exceed, bare/bear, cite/sight など) は全て発音記号の類似度が 0.5 以上であった．他の英語文章で，図 1 のアルゴリズムに基づき，類似度 0.8 以上で絞り込みを行った結果，平均 431 単語中 55 語(13%)の誤り候補数となり良い結果を示した．

5. まとめ

「誤りやすい英単語を認識させる手法」において，「誤りやすい語」の辞書を作成するための文字列の「似ている規則」について Hamming distance の採用にいたった理由を従来手法との比較を行った上で説明した．Hamming distance に対し，仮文字を使用するといった新しい概念手法を採用することによって，簡潔さと拡張性の双方を備える手法を提案した．この手法は，似ている語は，近傍の文字が異なる場合が多いという実態にもあった手法である．今後の課題として，精度を向上させるための LCS の利用方法を検討したい．

参考文献

[1]「誤りやすい英単語を認識させる手法」（菅野，金子，青木，情報処理学会第 67 回全国大会）

[2] Common Errors in English by Paul Brians
http://www.wsu.edu:8080/~brians/errors/errors.html#a

[3] Gonzalo Navarro, "A guided tour to approximate string matching", ACM Computing Surveys, Volume 33 Issue 1 (2001).

[4]Perldoc.jp project
http://bal4u.dip.jp/mt/program/archives/2004/12/_string_distanc.html