グルー検出を元にした辞書を用いない英文エラーの検出

6M-1

塩野谷 友隆† 梅村 恭司†
† 豊橋技術科学大学

1. はじめに

英文に混入する英単語でないアルファベットの文字列を非英語と呼ぶ. 非英語は機械翻訳やデータマイニング処理の精度を低下させる原因となるため, 非英語を抽出する研究が進められている. もし人間がこのような非英語を見たとき, その部分は他とは異なって見える. これは人間の中に，同じアルファベットで構成された語でも非英語かどうか検出できるエラー検出アルゴリズムがあるからである. そのようなアルゴリズムを計算機上で実装することが出来ればこのような非英語を機械的にエラーとして検出することが可能になる. そのためにはその英語と非英語の差異を定式化する必要がある. 我々は他の部分と比べて “稀” であることが英語と非英語を区別するものでは無いかと考え,“稀である” ということを定式化することにした. 本研究では出現頻度と文字列長に基づく尺度を用いて英語と非英語を区別する方法を提案する.

2. グルー

2.1 グルーとは

”稀である”ということの検出のアプローチとしてグルーというものがある. 稀, すなわち他に現れないというものの定義はいろいろ考えられるが, ここではパターンとパターンの隙間をグルーとして定義する. パターンとは反復して現れる文字列のことである. グルーの例を図 1 に示す. 図 1 のように

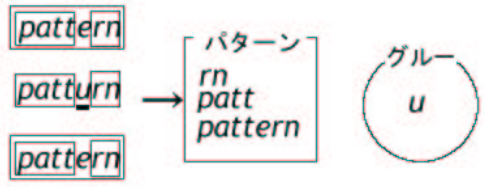

図 1 グルーの例 (矩形で囲ってある部分がパターン)
Fig. 1 An example of glue.(The part framed by square is pattern.)

エラーでパターンが分断された場合にグルーが発生する. 全てのグルーがエラーではないが少なくともエラーに関する情報が凝縮されたものである.

2.2 グルー検出の流れ

ある文字列を “パターン” と判定するためにはその文字列長と出現頻度を用いる. この二つの統計量を選択した理由は統計的なデータマイニングの定石であることと, なおかつ “ある文字列が複数回現れる” というパターンの定義に基づいた尺度であるからである. グルーを検出することがスペルミスなどの検出に役立つことが報告されており, ここでも同様に文字列長, 出現頻度を元にグルー検出を実装している.[2]. グルー検出は

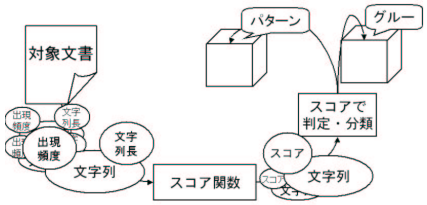

図 2 グルー検出の流れ
Fig. 2 The flow chart of detection of glue.

- 全ての部分文字列の文字列長, 出現頻度を計算する.
- その二つの統計量を入力とするスコア関数によりその文字列にスコアをつける.
- スコアが閾値より低い文字列をグルーとする.

という流れで実装される. これを図 2 に示す.

ところで本研究では単語自体のスペルミスではなく, エラーとなる非英語を検出することを目的としている. そこでグルー検出を単語単位で行うように拡張した.

2.3 スコア関数

プログラム言語などの形式言語では出現頻度, 文字列長がそれぞれ大きいパターンを形成することが多いが, 英文のような文脈自由言語では形式言語ほど大きなパターンが形成されることが少ない. 従来のグルー検出ではパターンの長さ, 大きさに関して式 (1) のような線形のスコアをつけていた.

$$BScore(P,L,F)=\sum_{l=1}^{l=L}\sum_{f=1}^{f=F}S(P) \tag{1}$$

$$S(P)=\begin{cases}1 & (L(P)>l\&\&F(P)>f)\\ 0 & otherwise\end{cases}$$

P : パターン

ただし $L(P)$ はパターンの長さを $F(P)$ はパターンの出現頻度を返す関数, L, F はそれぞれ文字列長, 出現頻度の閾値を決定するパラメータである. このスコア関数はパターンをグルーとみなす文字列長, 出現頻度の閾値の設定で最高スコアを決めることができるため最高スコアとエラー検出の関連性を測定することができる. しかしスコアの差異が小さくなるためグルーと判定するスコアの閾値の決定が難しい. そこで大きいパターンには大きなスコアがあたるように文字列長と頻度の積をスコア関数とした. このようにパターンのスコアとして出現頻度と長さの関数を適用した例が他にも報告されている.[1)]

$$AScore(P)=L(P)*F(P) \tag{2}$$

これはパターンの “広さ” として考えることができる. そこでこのスコアのつけ方を AreaScoring と呼ぶ. また前述した閾値を用いるものを BoundaryScoring と呼び, 両スコア関数を比較することで AreaScoring の有効性を計測した.

3. 評価

実際に図 3 に示すテストデータからグルーを検出し、正解

An error detection method for English text by glue detection.

† Tomotaka Shionoya (noya@ss.ics.ac.jp)
† Kyoji Umemura (umemura@tutics.tut.ac.jp)
Toyohashi University of Technology(†)

I can chcp 437 email you copies of these articles if you can't find them at yourlibrary. I've been using Managing Your Money for several years, and I have several friends who use Quicken, though I've not used it myself. CONFIG.SYS My overall impression is

図 3 メールデータに非英語を挿入したテストデータの一部 (下線部が検出したい非英語)
Fig. 3 The sample of test data.

```
CONFIG.SYS a:\\country.sys
country=049 nlsfunc
chcp 437 Sub procTest()
```

図 4 非英語データの一部
Fig. 4 The sample of noises.

集合と照らし合わせて, 精度を F-尺度☆で評価する.

このテストデータは WindowsOS に関するメールに図 4 に示すような非英語を挿入して生成したものである. ところでグルーの検出は辞書を必要とせずグルー検出を施したい文字列のみで行うことができる. しかしながら短い文字列では語数が少ない上, プログラムのように少ない語彙で表現もされていない. そこで大きな英文テキストを付加し語数を増やすことで精度向上を図った. 付加する英文テキストは新聞のデータを用いた. メールと新聞のデータには直感的に見て関連性が薄い. そのため意図的にエラーでない語のスコアを上昇させる効果はなく単純に学習用データとして用いても差し支えないと考えた. このように異種のデータセットをコーパスとするアプローチは 3) でも用いられており, これによって検索精度を向上させた報告がある.

また対象コーパスが英文であることに関連して, 単語の比較において Stemming を施した場合と施さなかった場合について実験を行った. Stemming は英単語の語尾を無視することで時制, 単複などによる単語の語尾変化を吸収する手法で, データマイニングにおいてしばしば用いられる手法である. 最終的に 2 種類のスコア関数を用いて,Stemming を行った場合と行わなかった場合, 計 4 種類のアルゴリズムを用いてエラー検出を行いそれぞれ評価した.

3.1 関数による効果

各アルゴリズムのスコアの閾値の設定による精度の変化を F-尺度を用いて計測した. 図 5 にスコアの閾値に対する F-尺度をプロットしたものを示す. AreaScoring を施したものが BoundaryScoring を施したものよりも上に来ている. また Stemming を施すと性能がよくなることも分かる.

BoundaryScoring は局所的には良い精度を示すが, 変動が激しく, 調整がしにくいと言える. これらのことから AreaScoring が BoundaryScoring より優れているといえる.

☆ F-尺度:再現率 R と適合率 P の調和平均で, 両者が大きな値を取るとき大きい値を取る. システムの精度を評価する尺度として用いられる.

$$F = \frac{2}{\frac{1}{R} + \frac{1}{P}}$$

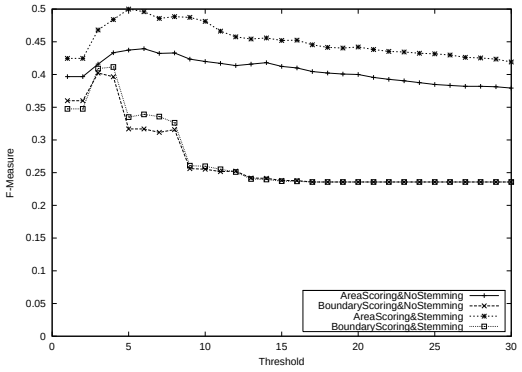

図 5 Score の閾値に対する F-尺度
Fig. 5 F-measure and threshold of score.

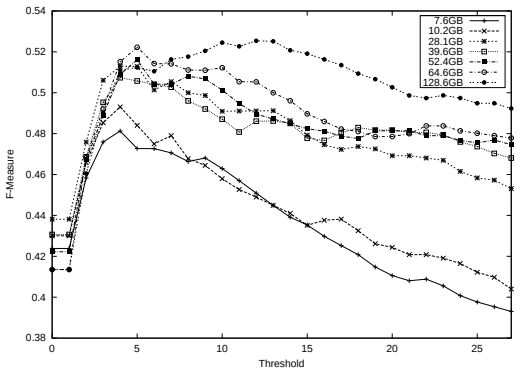

図 6 コーパスのサイズの変化に対する F-尺度の推移
Fig. 6 F-measures for various corpus size.

3.2 コーパスのサイズによる効果

付加するコーパスサイズを変化させ検出精度の変化を測定した. 図 6 にコーパスのサイズを変化させたときの F-尺度の推移を示す. アルゴリズムは最も精度の高かった Stemming を施した上で AreaScoring を用いたものを選択した. コーパスのサイズが大きくなるにつれて F-尺度が上昇した. またスコアの閾値の変化に対する F-尺度の変化が緩やかになりシステムとして安定したといえる.

4. ま と め

グルー検出アルゴリズムを単語単位で実装し, それを用いたエラー検出アルゴリズムを実装した. このアルゴリズムにおいて単語の比較には Stemming を施し, スコア関数にはパターンの"広さ"を用いることが検出精度の向上に繋がることを示した. また語彙を増やすためにコーパスを付加してエラー検出を行い, コーパスのサイズに応じて精度が向上し, システムとして安定することを示した.

参 考 文 献

1) 湯元紘彰, 森辰則, 中川浩志. 出現頻度と和接頻度に基づく専門用語抽出. **情報処理学会研究報告**, NL-147:111–118, 9 2001.
2) 吉川裕之, 貴島寿郎, 梅村恭司. n-gram 解析手法を応用したプログラムの欠損の検出. **情報処理学会論文誌**, 39:3294–3303, 12 1998.
3) 大井洋子, 大田佳宏, 今一修, 丹羽芳樹, 久光徹. 一般語とのあいまい性を持つたんぱく質名の自動検出. **情報処理学会研究報告**, NL-163:21–28, 9 2004.