1C-04

字種分割ツールの開発と公開
Development and release of division tool by character type

赤木　信也*
Shinya Akagi

*NTTデータ先端技術株式会社, Email: akagis@intellilink.co.jp

概要：日本語文は分かち書きされないため、文章解析では, 何らかの手法で文字を分割する必要がある. 現在では, MeCabを用いた統計的形態素分割を実施することが主流となっているが, 古くは, 字種分割という文字種単位の文字分割方法も検討されており, 可読性分野においては, 字種分割に基づいた可読性評価指標を開発する研究が存在している. しかし, 字種分割そのものに関する文献やツールは皆無であり, その有用性や課題については未整理となっている. そこで, 本研究では, 主に可読性分野に資することを目的として, 字種分割ツールの開発と公開を実施し, 有用性と課題の分析を行った結果を報告する.
キーワード：文章解析, 可読性, リーダビリティ, 字種分割

はじめに

日本語文は分かち書きされないため, 文章解析時には, 何らかの手法で文字を分割する必要がある. 現在では, 統計的形態素分割ツールであるMeCab[1]を使用して, 形態素単位に分割する手法が主流だが, 古くは, 字種分割という文字種単位に分割する手法も検討されており, 可読性分野においては, 字種分割に基づいた可読性評価指標を開発する研究が存在している[2]. 可読性分野については、人が文章を見た時の認知感覚が関わってくるため, 文字種単位の分割が有効に寄与しやすいと考えられる.
字種分割は, 古くから利用されているものの, それ自体の研究は皆無であり, 有用性や課題の整理が実施できていない. そこで, 本研究では, 主に可読性分野に資することを目的とし, 字種分割ツールの開発と公開を実施し, 有用性と課題の分析を実施する.

字種分割

字種分割とは, 文章を文字種単位（ひらがな, カタカナ, 漢字, 数字, アルファベット, 記号）に分割する手法である. 「今日の天気は晴れです。」という文章は, 「今日」「の」「天気」「は」「晴」「れです」「。」と分割される. 字種分割は, 分かち書きされる英文にも適用可能であり, 「Hello World!」という文章は, 「Hello」「 」「World」「!」と分割される.

字種分割の課題

字種分割は文字種単位で分割すれば良いため, 単純なように思えるが, いくつか課題が存在する.
一つ目は, 長音符（「ー」）とチルダ（「〜」）の扱いである. 長音符やチルダは, 記号として分割しても良いのだが, 実態としては, ひらがなの前後であればひらがな, カタカナの前後であればカタカナとして識別された方が妥当だと考える.
二つ目は, 数値におけるピリオド（「.」）とカンマ（「,」）の扱いである. ピリオドやカンマは記号であるため, そのまま記号として分割しても良いのだが, 実態としては, 「1.05」や「1,000」といった文章は, 全て数値として識別された方が妥当だと考える.
三つ目は, ピリオドやアンパサンド（「&」）を含んだ単語の扱いである. ピリオドやアンパサンドは記号であるため, そのまま記号として分割しても良いのだが, 実態としては, 「u.s.a」や「M&A」といった単語は, 全てアルファベットとして識別された方が妥当だと考える.
四つ目は, ピリオドで終わる単語の扱いである. ピリオドで終わる単語としては, 「u.s.」などがあるが, 単語として識別されるのが妥当であるが, コンピュータに認識させる場合に, 単語としてのピリオドか文末としてのピリオドかを単純には判定できないという課題がある.

前方記憶法

前方記憶法とは, 文章を文字単位で分割し, 前方の文字の文字種を記憶した上で, 現在の文字の文字種を決定する手法である. 前方記憶法を採用することにより, 前述した字種分割の課題の一つ目を改善することができる.

中間結合法

中間結合法とは, 文章を文字単位で分割し, 前方の文字の文字種を記憶した上で, 現在の文字が任意の文字で

あった場合に, 後方の文字種を判定して, 前方と中間と後方の文字を結合する手法である. 中間結合法を採用することにより, 前述した字種分割の課題の二つ目と三つ目を改善することができる. また, 課題の四つ目についても, 前方が特定の文字(u.sなど)だった場合に中間のピリオドのみを結合する(u.s.などにする)という処理を実施する（後方の文字は結合しない処理を実施する）ことで, 一部のピリオドで終わる単語を一つの単語として分割することができる.

ツールの公開

前方記憶法と中間結合法を採用した字種分割ツールを開発し, GithubおよびPyPIにツールを公開した[3]. 開発言語はpythonであり, Python Software Foundation Licenseで公開しているため, 改変や再配布の自由度を確保している. インストールは, python3をインストールした環境で, 「pip3 install divide-char-type」を実行すればよい.
可読性分析用に開発していた名残から, 関数の戻り値に文字種ごとの連なり数が返ってくるようになっており, 建石の可読性評価指標を再実装することが容易になっている. ただし, 「u.s.」という単語はアルファベットの連なり数2として計算され, ピリオドなどの結合文字が連なり数の計算に使用されない点は注意が必要である.

字種分割例

「こーどの鳥さん」という文章は, 「こーどの」「鳥」「さん」に分割される.「あなうさピーターの話」という文章は, 「あなうさ」「ピーター」「の」「話」に分割される.「1.05 is number」という文章は, 「1.05」「 」「is」「 」「number」に分割される. 「1,000 is number」という文章は, 「1,000」「 」「is」「 」「number」に分割される.「u.s. is state.」という文章は. 「u.s.」「 」「is」「 」「state」「.」に分割される.「I go to u.s.a. ok?」という文章は, 「I」「 」「go」「 」「to」「 」「u.s.a.」「 」「ok」「?」に分割される.

実行速度

1000字程度の文章, 憲法前文, ランダムな10000字について, mecab-python3を用いた形態素解析の実行速度と字種分割ツールを用いた字種分割の実行速度の比較を実施した. 実行環境は, MacBook Air, 1.7 GHz デュアルコアIntel Core i7, 8 GB 1600 MHz DDR3であり, Python 3.10.6を使用して計測した. mecab-python3の仕様なのか不明だが, 初めて計測した1回目計測結果と連続した計測結果で実行速度が大幅に変わったため, 1回目の計測結果と連続した10回平均の計測結果を取得した.

	MeCab1回目	divide_char_type1回目	Mecab10回平均	divide_char_type10回平均
1000字文章	0.251	0.014	0.003	0.013
憲法前文	0.058	0.001	0.002	0.001
ランダム10000字	0.132	0.017	0.011	0.017

計測結果から, 文章によっては字種分割の方が早く, 遅い文章でも３倍程度の差なので, 実行速度上は遜色なく分割処理を実施できることが示された.

今後の課題

文字種の範囲については議論の余地があり, 開発したツールでは, 漢字の範囲を一〜龠および々としているが, Unicode SetのScript=hanなどをベースにして, さらに多くの文字に対応させても良いと考える.また, 開発ツールでは, 半角ハイフン（「-」）を記号として分割しており、前後の文字種を見てカタカナとして処理することができていないため, 中間結合法を応用して対処する必要があると考える. 更には, ピリオドで終わる単語の網羅性を確保できていないため, Wikipediaなどの辞書からピリオドで終わる単語を抽出し, 中間結合を実施する対象の網羅性を確保すべきだと考える.

まとめ

字種分割ツールdivide-char-typeを開発および公開した. mecab-pythonとの処理速度の比較を実施した結果, 実行速度上は遜色なく分割処理を実施できることが示された.

参考文献

[1] MeCab: https://taku910.github.io/mecab/
[2] 建石由香, 小野芳彦, 山田尚彦: 日本文の読みやすさの評価式, 情報処理学会研究報告ヒューマンコンピュータインタラクション(HCI), Vol.25, pp.1-8, (1988).
[3] divide-char-type: https://github.com/ShinyaAkagil/divide_character_type