Wikipedia日本語版をコーパスとして用いた言語研究の手法

長谷部　陽一郎

1．はじめに

近年、コーパスを用いた言語研究の重要性が盛んに論じられている。[1] 英語圏には、以前よりBritish National Corpusを始め、いくつかの大規模コーパスが存在している。また日本でも、国立国語研究所の主導による大規模な書き言葉コーパスの構築が進められている。[2] しかし、計算機の高性能化や低価格化によって、処理の速度や環境の整え易さが向上した一方で、研究者が自由に使うことのできるコーパスの数や規模は限られている。より多くのコーパスの構築が望まれるが、コーパス構築にあたっては、テキストの著作権保護が常に大きな問題となる。

そのような中、インターネット上には、著作権フリーで利用できる一種の大規模日本語コーパスが存在する。Wikipedia日本語版である。[3] 日本語版を含めWikipediaには一般的に次のような特徴と、言語の研究者にとっての利点がある。第一に、すべての記事テキストを一定の条件のもとに誰もが自由に使用できる。そのため、同じ研究資源を異なる研究者やプロジェクト間で共有することが可能となる。第二に、多数の利用者によって執筆された数々の記事を含んでいる。このことは言語的バラエティの観点から非常に興味深い。第三に、定期的に全データが圧縮アーカイブファイルとして公開される。そのため、ある種の通時的な言語分析にも資することが期待できる。

本稿では、このようなWikipedia日本語版をコーパスとして用いた言語研究の可能性を探るとともに、Wikipediaの記事テキストを処理するために筆者が開発したツールキットの解説を行う。このツールキットを用いることで、

『言語文化』9-2：373－403ページ　2006.

圧縮アーカイブファイル内のデータを、コーパス処理に適した形式に変換することができる。また、Wikipediaの膨大な記事データの中から、指定した形態素パターンに合致する文字列の抽出を行うことができる。例えば、「このツールは言語分析にかかる時間と労力を省く」といった文字列の一部を抽出するのに、〈時間と労力を省く〉のような表層形式の指定だけでなく、〈名詞＋助詞＋名詞＋助詞＋動詞〉のような品詞の並びによる指定や、〈時間と労力＋助詞＋動詞〉といったミックス形式での指定が可能になる。

本稿の構成は次の通りである。２節ではコーパスとしてのWikipediaの特徴について論じる。Wikipediaの成り立ちや記事テキストの性質を概観した後、アーカイブデータの取得方法について述べる。３節では、Wikipediaを言語研究に活用するためのツールキットの概要を示す。具体的には、設計の方針と機能、使用する外部ソフトウェアなどについて述べる。４節では、ツールキットを構成する２つのRubyスクリプトwp2txt.rbおよびmconc.rbについて解説する。使用方法や入出力データの形式を示すとともに、これらのスクリプトの制限や課題についても論じる。

Wikipedia日本語版を活用することにより、最低限の環境を整えるだけで、用例採取や言語現象の定量的分析のための大規模コーパスが得られる。また、同一の言語データを異なる研究者やプロジェクト間で共有することができる。つまり、Wikipediaコーパスは、追試・修正・拡張・応用といった試みに対し、完全に開かれた研究資源を提供するのである。このことは、日本語を対象とする様々な言語研究の可能性を大きく広げると考えられる。

２．コーパスとしてのWikipedia日本語版

２.１ Wikipedia日本語版の概要

Wikipediaとは、インターネット上で誰もが閲覧・執筆・編集できるオンライン百科事典である。2001年に英語版のプロジェクトが開始され、2006年現在、200以上の言語により構成されている。そのうち、日本語を含む11の言語において、総記事数が10万件を超えている。英語版に至っては130万件超の記事を有しており、今や比類ない規模の情報集積体となっている。[4]

Wikipedia日本語版は、英語版の開始後、程なくして立ち上げられた。

2006年９月の時点で、保有記事は25万件を超えており、出版されている各種辞典類に比肩する規模となっている。英語版には及ばないものの、日本語版の利用ユーザ数は急激に増加していることから、今後はこれまで以上に記事数が増加していくものとみられる。[5]

WikipediaはMediaWikiというPHP言語によるプログラムを用いて運営されている。MediaWikiはWikiと呼ばれるウェブサイト構築システムの１つである。サーバサイドで稼動し、アクセスはインターネットに接続した端末からウェブブラウザを用いて行う。このため、利用者が特別なソフトウェアを用意する必要はなく、同時並行的な利用が可能である。また、テキストの加筆・編集の際にWiki構文と呼ばれる一定の規則を用いるのも特徴である。Wiki構文は、ウェブページ記述の標準形式であるHTMLをサーバサイドで生成するための簡便記法で、HTMLよりもシンプルに記述でき、他項目や外部ページに対するリンクも簡単に設定できる。

Wikipediaはフリーの百科事典であるが、当然ながら、すべての意味において「フリー」という訳ではない。利用者はWikipediaのテキストが「著作権フリー」であることの意味を正しく理解しておく必要がある。Wikipediaのテキストは GNU Free Documentation License (GFDL) に基づいて使用が許諾されている。[6] GFDLは、知的生産物の共有と創造の促進を目指す非営利団体Free Software Foundationにより配布されているライセンス形態である。[7] 以下にGFDLの概要を示す。

GFDL文書は無断で改変・再配布・二次利用が可能である。
ただし、二次著作物もまたGFDLの条件を継承する。

当然ながら、他人が著作権を保有するコンテンツをWikipediaの記事に無断転載することはできない。このことは、きわめて重要である。なぜならGFDLのライセンスは、単にその文書に適用されるばかりでなく、それに基づいたあらゆるコンテンツに波及するからである。Wikipediaでは、記述に問題が発見された場合、一定の手続きを経た後、記事削除などの処置が行われる。[8]

2.2 言語データの特徴

印刷物として出版されている百科事典同様、Wikipediaの各項目の記事は、客観的な視点から書かれた説明文が主体となっている。もちろん、不特定多数の利用者が執筆や編集に関わることから、事実の捉え方や記述の様式が、部分によって異なってくることは避けがたい。しかし、明らかにバイアスのかかった意見の表明や、極端に個性の強い文体の使用を避けるべきことは、大部分の利用者に前提として共有されている。このことから、Wikipedia日本語版の文体は基本的に、標準的な日本語の書き言葉のそれであると言える。[9]

多数の利用者が執筆者・編集者となっていることは、一方で日本語コーパスとしてのWikipediaを特徴的で興味深いものにしている。類似した構成の記事であっても、実際の語彙・表現・文体は多かれ少なかれ異なる。このことは、言語研究の観点からすると歓迎すべきことである。ある特定の人物や組織の言語観によらない、一般の人々の手によるテキストは、規範性についてはともかく、現代日本語話者の平均的な言語感覚を反映している可能性が高い。

また、扱われる内容が、通常の百科事典のそれに限定されないことも注目に値する。Wikipediaでは、新しい文化、事件、概念、商品などが、項目として積極的に取り入れられている。これらの記述においては、必然的に新しい語彙や表現が多用される。また、既存の表現でも、慣例から幾分逸脱した形で用いられることがある。従来、新語や新用法は、時を経て日本語に定着しない限り、資料として残りにくいものであった。しかしWikipediaでは、全データが定期的にアーカイブされる。研究者は必要に応じてそれらにアクセスすることができる。データが十分に蓄積された後には、アーカイブを時系列順に辿り、言葉の意味や用法が変化する過程を観察・分析することも可能になるだろう。

2.3 インターネット上のテキストの問題点

近年、理論言語学や言語教育を含む、ことばに関連する多くの領域で、インターネット上のテキストをコーパスとして利用することが注目されている

（鷹家・須賀1998; 斉藤・赤野・中村2005）。しかし、インターネット上のテキストには、次のような問題が存在する。

まず、研究者間で完全に同一のデータセットを共有することが難しいという事実がある。インターネット上で検索を行う際、GoogleやYahoo!などの全文検索型のサーチエンジンが一般的に用いられる。しかし、これらのサーチエンジンのデータベースは随時更新されるため、検索結果の再現性は低い。データの定量的な面が関わってくるとき、このことが特に問題となる。また、それぞれのサーチエンジンが独自のアルゴリズムを用いていることや、それらが必ずしも公開されていないことは、データのバランスや信頼性を下げる要因となっている。

さらに、インターネット上の言語データは必ずしも現実の言語事実を反映していないという問題がある。例えば大規模掲示板サイトなどでは、明らかに独自の言語空間が形成されている。また一部の企業サイトでは、いわゆるSEO (Search Engine Optimization) のためのデータ操作が行われている。例えば、自社サイトの検索ヒット率を上げるため、HTMLに「見えない」テキストを埋め込むという手法がある。これらは、サーチエンジンでの検索結果を言語研究に利用する際、非常に不都合な要素となる。

同じインターネット上のデータでも、Wikipediaをコーパスとして用いるときに、これらのことが問題になる可能性は低い。もちろん、注意すべき点は存在する。例えば、誰もが記事の執筆・修正に参加できることから、特定の個人や組織が自らの利益のためにWikipediaのデータを操作する可能性がある。事実、記事の一部を恣意的に削除・修正した利用者がそのことを非難されるという事例が、Wikipedia英語版でも、日本語版でも発生している。また、インターネット上のプロジェクトであるという性質上、特定のカテゴリーに属する記事（コンピュータ関連やサブカルチャー関連）の比率が高いという傾向が存在する。しかし、Wikipediaはまだ開始されて間もないプロジェクトである。急速に発展してきたとはいえ、現在のシステムが立ち上がってから、まだ数年にしかならない。今後、その存在と理念が多くの利用者の中で共有され、定着していくならば、上のような問題点も多かれ少なかれ改善されていくと思われる。

2.4 アーカイブデータについて

Wikipediaのデータベースダウンロードページでは、各種アーカイブデータの圧縮ファイルが公開されている。[10] ファイルの生成は通常、月に１度以上行われている。ダウンロードサイトでは様々な内容のファイルが提供されており、編集の履歴に至るまで、すべてのデータを含んだ巨大なファイルもあれば、リンクデータのみを集めた比較的小さなファイルもある。これらの中で日本語のコーパス研究に最も有用なのは、全記事データと主要なメタデータを含んだ、次のような形式のファイルである（数字はデータ生成の日付を表す）。

jawiki-20060803-pages-articles.xml.bz2

次の３節および４節では、このファイルを言語研究にどのように役立てられるかについて見ていく。

3．Wikipediaコーパス処理用ツールキットの概要

本節では、Wikipedia日本語版のアーカイブデータを用いてコーパス研究を行うためのツールキットの概要を示す。このツールキットは２つのスクリプトから構成されており、以下では、まずその設計方針について述べる。次に、必要となる外部ソフトウェアについて解説する。なお、ツールキットの最新バージョンは以下の筆者ウェブサイトからダウンロードできる。

http://www.yohasebe.com/software/

3.1 ツールキットの設計方針と実行形態

WikipediaのアーカイブデータはbZ2圧縮されたXMLファイルとして配布されており、各記事のタイトルがXMLツリーのtitle要素内に、本文データがtext要素内にそれぞれ格納されている。そこで、ツールキットを構成する第1のスクリプト (wp2txt.rb) には、bz2を展開しつつXMLデータをスキャンして

title要素およびtext要素内のデータを抽出する機能を実装する。また、このスクリプトには、text要素中の記事データから不要なWikiタグおよびHTMLタグを除去し、扱いやすい日本語テキストに変換する機能を加える。ただし、タグの中には、フォントの強調や、他のページへのリンク設定など、テキストの内容自体と直接関係がないもの以外に、項目の見出しやリスト表記の印など、テキストの論理構造に関連するものが存在している。これら論理構造タグについては、記事データの可読性を保つため、デフォルトでは除去を行わない。

テキスト抽出の際には、「文」をコーパスデータの主要な単位とし、記事タイトルや項目の見出しを除いて、１個以上の句点（。）を持った行のみを有効な言語データとする。[11] 句点を持たない（すなわち文を含まない）行はテキスト抽出から除外する。もちろん、これによりテキストの取りこぼしが生じる可能性は否定できない。しかし、句点を持たない文字列を抽出対象にすることによって増えるノイズデータの量を考えたならば、現実的で妥当な方法であると言える。

次に、第２のスクリプト (mconc.rb) についてであるが、こちらには、上記の処理によって取り出されたコーパスデータから、指定した形態素パターンに合致する文字列を取り出す機能を実装する。具体的な処理は次の通りである。まず記事データを読み込んで文の単位に分解する。次に形態素解析にかける。そして出力結果を、設定ファイルにあらかじめ記述された正規表現パターンと照らし合わせ、データの取捨選択を行う。最後に、抽出された文字列と、それらの詳細な形態素情報とをCSV (Comma Separated Value) ファイルとして出力する。

各スクリプトの役割と関係を図式化すると、図１のようになる。

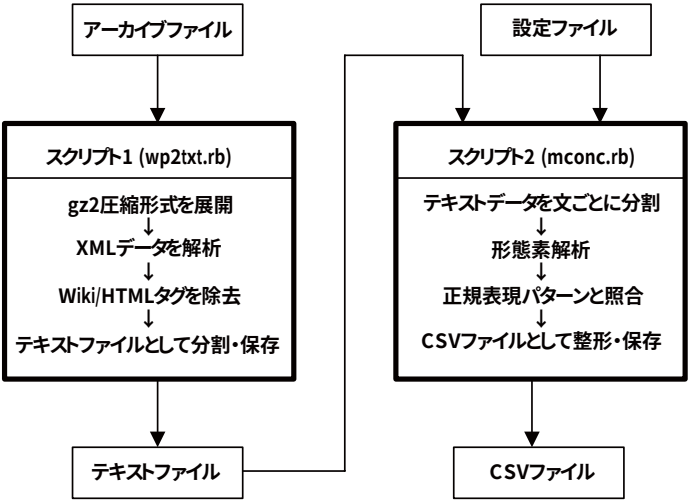


図1

なお、wp2txt.rbとmconc.rbは、いずれもRubyによるコマンドライン・アプリケーションとして作成する。したがって、これらの実行は、コンソール（Windowsの場合はコマンドプロンプト）上で、コマンドを対象ファイルおよび各種オプションとともに入力する形になる。

3.2 使用する外部ソフトウェア

ツールキットの開発には、次に挙げるようなオープンソース・ソフトウェアを利用する。各スクリプトを使用する際にも、実行環境にこれらがインストールされている必要がある。[12]

3.2.1 Ruby

Rubyは、まつもとゆきひろ氏によって開発されている日本発のプログラミング言語である。[13] テキスト処理やウェブアプリケーションを始めとする様々な分野で使用されており、特に近年はRubyで開発されたウェブアプリケーション・フレームワークRuby on Railsの人気もあって、世界中で注目さ

れている。

Rubyの特長としては、次のような点が挙げられる。

１．テキスト処理に優れている。
２．シンプルな文法を持っている。
３．スクリプト言語であり、移植性が高い。

１は最も重要な点である。Rubyには正規表現機能が組み込まれており、従来広く用いられてきたPerlに劣らないテキスト処理能力を持っている。また、多言語に対応しており、日本語テキストの扱いも問題ない。文字コードとしてShift JIS、EUC、UTF-8がいずれも指定可能であり、相互変換も容易である。WikipediaのデータはUTF-8で記述されているが、環境によっては、後の処理のために別の文字コードに変換する必要がある。Rubyのテキスト処理機能は、そのようなニーズにも柔軟に対応することができる。

２は、スクリプトを実際に作成していく上で大きな利点となる。文法がシンプルであるうえ、ライブラリモジュールが豊富に用意されているので、必要な機能を短時間で実装することができる。コーパス分析の際に求められる処理は、それぞれのプロジェクトによって大きく異なる。したがって、コーパス処理のためのツールキットは改良・拡張しやすいことが望まれる。Rubyによるコードは可読性が高いことで知られており、この点においても今回の目的に適っている。

また、３にあるようにスクリプト言語であるRubyで書かれたプログラムは、実行のためにコンパイルを行う必要がない。Rubyの実行環境さえ整っていれば、WindowsやLinuxといったプラットフォームの別に関係なく、即座に実行することができる。

Rubyには以上のような特長がある一方で、実行速度において他の言語に比べ若干劣るという欠点がある。しかし、近年は計算機の性能が向上し、プログラムの速度自体はそれほど大きな問題ではなくなった。コーパス分析においては、しばしば巨大なサイズのデータを扱うため、高速であるに越したことはない。それでも、必要な機能を効率良く実装できることや、コードの

可読性が高いことは、多くの場合、目的の達成に必要な作業コストを大幅に削減してくれる。

Rubyの開発および実行環境は、以下の公式ウェブサイトからダウンロードできる。[14]

http://www.ruby-lang.org/ja/downloads/

3.2.2 bz2

bz2は、効率の良いファイル圧縮形式の一つであり、近年、UNIX系のオペレーティングシステム上で盛んに用いられている。すでに触れたとおり、Wikipediaでは、XML形式で記述されたファイルをbz2形式で圧縮したものをアーカイブデータとして配布している。

bz2形式のファイルを展開するためのコマンドは、Linuxなどのシステムにはあらかじめ含まれていることが多い。その場合、新たにプログラムをインストールする必要はない。wp2txt.rbはコマンドラインにパイプをつなぎ、bz2コマンドを呼び出してファイルを展開しながらテキスト処理を行う。そのため、実際に処理を行う際には、圧縮ファイルの展開を意識することもない。

もしbz2コマンドが利用可能でない場合は、パスの通ったディレクトリにプログラムをインストールする必要がある。各プラットフォーム用の実行ファイルは、以下のサイトからダウンロード可能である。

http://www.bzip.org/downloads.html

3.2.3 MeCab

MeCab（和布蕪）は、工藤拓氏が開発を進めている形態素解析システムである。[15] 効率的なアルゴリズムを採用しており、高い解析精度を実現しながら、他の形態素解析システムに比べて高速に動作する。[16]

LinuxなどのオペレーティングシステムでMeCabを利用するためには、プ

ログラム本体とともに専用の辞書をインストールし、設定を行う必要がある。一方、Windows版のMeCabには辞書が同梱されており、特に設定を行う必要はない。いずれも、次のサイトからダウンロードできる。

http://mecab.sourceforge.jp/

使用可能な辞書ファイルはいくつか用意されているが、IPA辞書の使用が推奨されている。IPA辞書とは、情報処理振興事業協会 (IPA) によるIPA品詞体系に基づいた、日本語の形態素情報データベースである。本稿執筆時、MeCabの最新バージョンは0.93、IPA辞書の最新バージョンは2.70であった。Wikipedia用ツールキットの作成にもこれらを利用した。

ここで、MeCabがどのような働きをするかについて簡単に見ておく。コマンドラインからMeCabを起動すると、日本語のテキストを受け付ける状態になるので、例として次のような文を入力する。

貴社の記者が汽車で帰社した。

すると、例１のような解析結果が示される。

```
貴社	名詞,一般,＊,＊,＊,＊,貴社,キシャ,キシャ
の	助詞,連体化,＊,＊,＊,＊,の,ノ,ノ
記者	名詞,一般,＊,＊,＊,＊,記者,キシャ,キシャ
が	助詞,格助詞,一般,＊,＊,＊,が,ガ,ガ
汽車	名詞,一般,＊,＊,＊,＊,汽車,キシャ,キシャ
で	助詞,格助詞,一般,＊,＊,＊,で,デ,デ
帰社	名詞,サ変接続,＊,＊,＊,＊,帰社,キシャ,キシャ
し	動詞,自立,＊,＊,サ変・スル,連用形,する,シ,シ
た	助動詞,＊,＊,＊,特殊・タ,基本形,た,タ,タ
。	記号,句点,＊,＊,＊,＊,。,。,。
EOS
```

例１

これは次のようなMeCabの標準フォーマットに基づいた出力である。

表層形　品詞,細分類1,細分類2,細分類3,活用形,活用型,原型,読み,発音

ツールキットでは、MeCabによるこのような出力をスクリプト内部から取得し、表層形とその他の形態素情報のそれぞれを、設定ファイルに記述された正規表現と照合する。これにより、特定のパターンに合致する文字列をコーパスから抽出し、詳細な情報と共に得ることが可能になる。

4．各スクリプトの詳細

本節では、ツールキットを構成するスクリプト、wp2txt.rbとmconc.rbの使用方法および入出力データの仕様を示す。また、実装上の制限や使用上の注意点についても述べる。

4.1 wp2txt.rb

wp2txt.rb (Wikipedia to text) はその名の通り、Wikipediaのアーカイブデータを圧縮アーカイブファイルから抽出し、コーパスとして扱いやすいプレーンなテキスト文書に変換するためのスクリプトである。

4.1.1 使用方法

Wikipediaの圧縮アーカイブファイルをダウンロードした後（２節参照)、コンソール（コマンドプロンプト）に次のような書式でコマンドを入力し、実行する。

>ruby wp2txt.rb 対象ファイル名 [オプション]

環境にもよるが、通常は数分から数十分の作業の後、テキストファイルが所定のディレクトリに出力される。以下は指定可能なオプションの例である。

-o [dir] --output-dir=[dir]

出力ファイルの保存先ディレクトリをdirに設定する。

-s [size] --file-size=[size]

各出力ファイルのサイズが、およそsize (MB) になるように分割する（デフォルトは10MB）。

-e [code] --output-encoding=[code]

出力ファイルの文字コードをcodeに指定する。UTF8、SJIS、EUCのいずれかが指定可能（デフォルトはUTF8）。

-n --no-conversion

データ変換をせず、XML形式を保ったままファイル分割だけを行う。

-h --skip-heading

Wiki記法で見出しとして指定されている部分のテキストを抽出から除外する。

-l --skip-list

Wiki記法でリストとして指定されている部分のテキストを抽出から除外する。

4.1.2 wp2txt.rbの入出力データ

ここでは、wp2txtに入力されるXMLデータと、変換済みテキストデータの形式を確認していく。まず、アーカイブファイルに含まれるXMLデータは、下の例2のような形式になっている。

```
<page>
  <title>形態素</title>
  <id>2622</id>
  <revision>
    <id>6608356</id>
    <timestamp>2006-07-13T00:30:46Z</timestamp>
    <contributor>
      <username>Eskimbot</username>
      <id>29676</id>
```

```
    </contributor>
    <minor />
    <comment>robot  Adding: [[nds:Morphem]]</comment>
     <text xml:space="preserve">'''形態素'''（けいたいそ）とは、言語の中で意味を持つ最小単位である。形態素のうち、単独で[[語]]として現れ得るものを自由形態素、単独では用いられず他の形態素とともに現れるものを拘束形態素または束縛形態素という。

==概要==
拘束形態素は[[語形成]]上、[[派生]]を行う。例えば「反作用」という語は、拘束形態素である[[接辞]]の「反」と自由形態素の「作用」からなる派生語である。

一つの形態素が複数の現れ方をすることがある。これを異形態という。例えば「'''あめ'''ふり」、「'''あま'''やどり」、「きり'''さめ'''」に含まれる「あめ」、「あま」、「さめ」は同じ形態素「あめ」の異形態である。

（後半は省略）
```

例２（記事「形態素」より）

各記事データの1行目では、pageタグにより記事の開始が示される。その後いくつかのメタ情報タグと共に、textタグと記事本文が現れる。記事本文のテキスト中に見られる記号の多くは、フォント装飾やリンク設定を施すためのWikiタグである。例えば、'''形態素''' のように、２個以上のアポストロフィでマークされた文字列は、アポストロフィの個数に応じて強調表示されることになっている。また、[[語形成]] のように二重の角括弧でマークされた文字列は、他の項目へのリンクとして機能する。

例２のようなXMLデータを含んだファイルをwp2txt.rbに入力し、デフォルトの設定で実行すると、例３のような形式に変換される。

[[形態素]]
形態素（けいたいそ）とは、言語の中で意味を持つ最小単位である。形態素のうち、単独で語として現れ得るものを自由形態素、単独では用いられず他の形態素とともに現れるものを拘束形態素または束縛形態素という。
==概要==
拘束形態素は語形成上、派生を行う。例えば「反作用」という語は、拘束形態素である接辞の「反」と自由形態素の「作用」からなる派生語である。
一つの形態素が複数の現れ方をすることがある。これを異形態という。例えば「あめふり」、「あまやどり」、「きりさめ」に含まれる「あめ」、「あま」、「さめ」は同じ形態素「あめ」の異形態である。

例３

例３では、メタ情報を示すタグや、文字装飾およびリンク設定タグがすべて削除されている。また、例２のXML文書においてtitleタグによってマークされていた記事タイトルは、二重角括弧で囲まれた形で表示されている。これは、後でさらにデータを処理する際に、タイトルの把握を容易にするためである。本文中の二重角括弧はすべて取り除かれることになっているため、記事タイトルと本文とは確実に分離される。なお、記事テキストの可読性を保つため、見出しを表す等号タグ (==) は残してある。もちろん、後処理の段階で簡単に読み飛ばすことができる。

４.１.３ wp2txt.rbによるデータ変換の注意点

ここで、wp2txt.rbを用いたデータ変換に関するいくつかの注意点について述べておきたい。まず、wp2txt.rbは、元のアーカイブデータに含まれるすべての日本語テキストを抽出するわけではない。言語研究用コーパスの構築という目的を最優先し、後の過程で「ノイズ」とみなされる可能性の高い要素はこの段階で取り除かれる。具体的には次のような要素が抽出から除外される。

１．メタ的な機能を持つpage要素

文字列「Wikipedia:」で始まるpage要素内には、Wikipedia自体についての解説や編集履歴などが記載されている。これらは有効なテキストとみなさない。同様に、「Template:」「Category:」「画像:」で始まるタイトルのpage要素もテキスト抽出の対象としない。

２．削除提案などのメッセージ

記事の中には、削除提案がなされているものがある。該当するページでは、本文が表示されず、代わりに削除提案中である旨のメッセージが示される。アーカイブファイルに含まれるこれらのメッセージは抽出から除外する。また、他のメタメッセージについても同様である。

３．コメントタグの内部

Wikipediaの記法では、HTMLに準じたコメントタグの使用が認められている。記事の執筆者はこれを用いて、他の執筆者への連絡事項や、解決されていない問題に関するコメントを残すことができる。これらはWikipediaの閲覧用ページには現れないテキストであり、抽出の対象としない。

４．画像のキャプションおよび句点（。）を含まない文字列

Wikipediaの記事には、ウェブ上の百科事典という性質上、一般的な文章形式をとらない文字列が多く含まれている。例えば、画像のキャプションや、テーブル内の要素である。これらを含め、句点を含まないテキストは前述の通り抽出から除外する。

次に、注釈の扱いについて述べておく。Wikipediaでは、refタグでテキストをマークアップすることにより、脚注を本文中に埋め込むことができる。これらをそのまま残しておくと、後のテキスト分析の段階で何らかの特別な処理を行う必要が生じる。しかし、注釈部分はテキストとして高い価値を持つことも多く、完全に削除してしまうことは避けたい。そこで、<ref> と </ref> とは、対応する丸括弧に置き換えることにする。これにより、下の例４に示されるデータは、例５のような形に変換される。なお、オリジナルのXMLデータにおいて < と > は、それぞれ < と > というHTML文字実体参照コードで表現されており、wp2txtでは、これらを実際の文字に置き換え

た上で処理を行う。[17]

```
== 本割 ==
本場所における正規の[[取組]]を'''本割'''（ほんわり）と呼び、本割は番付・成績などを加味して審判部により決められる。幕内は昼過ぎ（初日のみ前々日、千秋楽は前日17:00ごろ）、十両は17:00ごろ、幕下以下は18:00以降に翌日の取組が発表される。かつては東西対抗戦、一門系統別総当たり制などで行われていたが、現在は部屋別総当たり制&lt;ref&gt;優勝決定戦を除いて、同じ部屋に所属する力士同士の対戦は組まれない。他にも不文律として同じ部屋に所属していなくても兄弟同士の取組は組まれない。つまり[[北桜英敏|北桜]] - [[豊桜保勝|豊桜]]、[[露鵬幸生|露鵬]] - [[白露山佑太|白露山]]の取組は本割では組まれないのである。&lt;/ref&gt;で行われる。
```

例4

```
== 本割 ==
本場所における正規の取組を本割（ほんわり）と呼び、本割は番付・成績などを加味して審判部により決められる。幕内は昼過ぎ（初日のみ前々日、千秋楽は前日17:00ごろ）、十両は17:00ごろ、幕下以下は18:00以降に翌日の取組が発表される。かつては東西対抗戦、一門系統別総当たり制などで行われていたが、現在は部屋別総当たり制（優勝決定戦を除いて、同じ部屋に所属する力士同士の対戦は組まれない。他にも不文律として同じ部屋に所属していなくても兄弟同士の取組は組まれない。つまり北桜 - 豊桜、露鵬 - 白露山の取組は本割では組まれないのである。）で行われる。
```

例5

Wikipediaでは、「本割り（ほんわり）」のように、重要語句の仮名表記を、丸括弧を用いて示すということが行われる。また、「昼過ぎ（初回のみ前々日…）」のように、解説を丸括弧の中に記述するということも行われている。これらはいずれも、後処理の中で何らかの処理を行う必要がある部分である。refタグによる注釈を同様に丸括弧で囲むことは、本文に対して付加的な意味合いを持つテキストの処理方法を共通化するという効果もある。

ところで、wp2txt.rbのような種類のスクリプトにデフォルトでどれだけの

処理をさせるかというのは、難しい問題である。通常、メタ的要素やタグ類はできるだけ取り除くことが望ましい。一方で、後処理の方法や目的によっては、これらの要素が役に立つこともあり得る。そこでwp2txt.rbでは、オプション設定で出力の形式をある程度調整できるようにしている。これにより、次に解説するmconc.rbと組み合わせて使用する他、目的によっては単体で使用することも可能となっている。

4.2 mconc.rb

mconc.rb (morphological concordancer) は、Wikipediaの記事の中から、指定した正規表現パターンに合致する文字列をすべて抽出し、詳細な形態素情報とともにCSVファイルに書き出すスクリプトである。

4.2.1 使用方法

設定ファイル（詳細は後述）に、正規表現を用いた文字列検索条件の記述を行い、コンソール（コマンドプロンプト）に次のような書式でコマンドを入力し、実行する。[18]

```
>ruby mconc.rb 対象ファイル名 [オプション]
```

対象ファイル名には、wp2txt.rbの実行によって得られたテキストファイルを指定する。あるいは、*.txtといったグロブ表記を用いて、複数のファイルを順次処理していくこともできる。オプション指定なしで実行した場合は、mconc.rbと同じディレクトリに保存された設定ファイル (config.yaml) を読み込み、入力ファイルと同じディレクトリに出力ファイルを保存する。指定可能なオプションには次のようなものがある。

-o [dir] --output-dir=[dir]

出力ファイルの保存先ディレクトリをdirに指定する。

-c [file] --config-file=[file]

設定ファイルの場所を指定する。

-e [code] --output-encoding=[code]

出力ファイルの文字コードをcodeに指定する。UTF8、SJIS、EUCのいずれかが指定可能（デフォルトはSJIS）。

-a --skip-annotation

丸括弧で囲まれた注釈テキストを読み飛ばす。

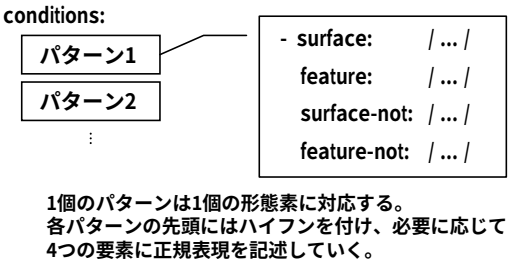
-s --with-sentence

オリジナルの文を出力に含める（注釈の扱いは-aオプションに従う）。

4.2.2 設定ファイルの記法

mconc.rbの実行時に読み込まれる設定ファイルには、文字列抽出のための条件や各種の設定項目をYAML形式で記述する。[19] YAMLとは、様々なデータを連続化 (serialize) し、テキストファイルとして保存するためのデータ形式である。XMLと同様の機能を持つが、XMLが仕様上の制約により可読性が犠牲になっているのに対し、YAMLは人間にとっても読みやすい形式を備えている。今回の目的に適したデータ形式であると言える。

設定ファイルに記述する文字列抽出条件の書式は次のようになっている。



１行目のconditions: は、以下に抽出条件が示されることの宣言である。[20] この次の行以降では、YAMLの規定に従って2個のスペースで字下げをし、マッチを行いたい形態素の数だけパターンを記述していく。

各パターンは１個のハイフンによって導かれ、surface、feature、surface-not、feature-notという４つの要素から構成される。このうち中心的な役割を

果たすのはsurfaceとfeatureである。surfaceでは、表層形、すなわち出現する形態素の文字通りの形にマッチする正規表現を指定する。featureでは、形態素の品詞や活用形などの情報にマッチする正規表現を指定する。これらはMeCabの標準出力フォーマットに対応している。以下にMeCabの標準出力の一例を示す。

```
落書きを消しなさい。
落書き  名詞,サ変接続,*,*,*,*,落書き,ラクガキ,ラクガキ
を      助詞,格助詞,一般,*,*,*,を,ヲ,ヲ
消し    動詞,自立,*,*,五段・サ行,連用形,消す,ケシ,ケシ
なさい  動詞,非自立,*,*,五段・ラ行特殊,命令ｉ,なさる,ナサイ,ナサイ
。      記号,句点,*,*,*,*,。,。,。
EOS
```

例６

この中で、「消し」という形態素に関する部分の構成は次のようになっている。

surface:	feature:								
表層形	品詞	細分類１	細分類２	細分類３	活用形	活用型	原型,	読み	発音
消し	動詞	自立	*	*	五段・サ行	連用形	消す	ケシ	ケシ

表１

MeCabがこのような出力を行うことから、例えばmconc.rbを用いて、「消し」という形態素をコーパスから抽出したいのであれば、次のように、表層形「消し」に合致する正規表現を、形態素パターンのsurface要素の部分に記述すればよい。

```
surface: /^消し$/
```

ここで、記号＾と＄は、それぞれ文字列の先頭と末尾を意味する正規表現で

ある。[21] これらを付けておくことで、「打ち消し」や「消しゴム」といった合成語の一部が誤って抽出されるのを防ぐことができる。

あるいは、特定の形態素でなく、動詞というカテゴリーに属する形態素を包括的に抽出したいならば、先頭に「動詞」を含む文字列に合致する正規表現を、feature要素として記述すればよい。

```
feature: /^動詞/
```

featureに含まれる形態素情報において品詞情報は先頭に記載されるので、ここでも文字列の先頭を意味する記号 ^ を用いている。しかし、品詞の後には常に他の情報が続くため、文字列の終端を表す記号 $ を付けてはならない。

設定ファイルに記述する形態素抽出パターンには、その他、surface-not、feature-notという要素を指定することができる。これらは、それぞれsurfaceとfeatureに対応した否定条件である。例えば、

```
surface-not: /^(こと¦もの)$/
```

と記述すると、「こと」「もの」という表層形を持った語が抽出から除外される。また、

```
feature-not: /^動詞/
```

と記述すると、動詞のカテゴリーに属するあらゆる形態素が除外される。

このように、形態素１個に対応するパターンを、surface、feature、surface-not、feature-notという４つの要素を組み合わせて構成していく。その際、パターンは各要素の論理積として評価される。また、これらの要素を一切指定せず、ハイフンのみを付けた行を設けると、あらゆる形態素とマッチするパターンを記述することができる。[22]

ここで、少し複雑な例を１つ挙げてみよう。

conditions:

- feature:	/^名詞/	形態素情報の最初に「名詞」を含む形態素
- feature: surface-not:	/^助詞/ /^(は|も)$/	形態素情報の最初に「助詞」を含み、かつ 表層形が「は」「も」でない形態素
-		あらゆる形態素
- surface: feature:	/^で$/ /^助動詞/	表層形が「で」で、かつ 形態素情報の最初に「助動詞」を含む形態素

この条件式を設定ファイルに記述して保存し、実際にmconc.rbを用いてWikipediaコーパスを検索したところ、「非常に複雑で」（記事「日本語」より）、「正方形の広場で」（記事「パリ」より）などの文字列が抽出された。

なお、設定ファイルには、文字列の抽出条件だけでなく、４.２.１で示したコマンドライン・オプションを記述して保存することができる。[23] また、出力されるCSVファイルのフォーマットを指定するなど、いくつかの用途がある。

４.２.３ mconc.rbの出力データ

ここでは、実際にmconc.rbを用いて文字列の抽出を行い、結果出力を確認する。対象コーパスとしては、2006年８月３日に生成されたWikipedia日本語版アーカイブデータ（bz2形式で圧縮された状態で約280MB）を用いる。

まず、前処理としてwp2txt.rbを用いた変換作業を行う。実行に際しては特にオプションを指定せず、デフォルトの処理を行う。結果として、約10MBのテキストファイルが54個生成された。次に、設定ファイル (config.yaml) に下のような記述を行う。

conditions:

- surface:	/^強制$/
- feature:	/^名詞/
- feature:	/^動詞.*サ変/
-	

この条件式は、

「強制」＋（動）名詞 ＋ サ変動詞 ＋ 何らかの形態素

という構文形式の文字列にマッチすることが期待される。[24]

ところで、小林 (2004) によると、日本語の複合動名詞には「〜する」（サ変動詞非使役形）をとることができるタイプと、「〜する」をとれず、「〜させる」（サ変動詞使役形）のみをとるタイプの2種類がある。小林 (2004) は前者を「強制排除タイプ」、後者を「強制着陸タイプ」と呼んでいる。上に示したような条件式を与えてmconc.rbを実行することにより、このような言語現象の実例と統計情報を容易に得ることができる。

実行に当たっては、-sオプションを用いて、オリジナルの文を結果出力に含める設定を行う。mconc.rbを起動し、計54個のテキストファイルすべてを処理させたところ、元のコーパスデータに含まれていた4,179,840文から、パターンに一致する381の文字列が抽出された。

出力されたCSVデータの一部が下の表２である。ここでは紙面の都合上、全381例の中から、「強制着陸」と「強制排除」を含んだ例をそれぞれ１つずつ示すにとどめる。[25]

	記事タイトル	ヒット文字列	オリジナル文
1	ガトリング砲	強制排除し、	…不発の実包が混入していても動力で強制排除し、…
2	ハイジャック	強制着陸させ	…同じ空軍基地に強制着陸させられた。

	表層形-1	品詞-1	細分類 1-1	活用形 1-1	活用型-1	原形-1
1	強制	名詞	サ変接続	*	*	強制
2	強制	名詞	サ変接続	*	*	強制

	表層形-2	品詞-2	細分類 1-2	活用形-2	活用型-2	原形-2
1	排除	名詞	サ変接続	*	*	排除
2	着陸	名詞	サ変接続	*	*	着陸

	表層形-3	品詞-3	細分類 1-3	活用形-3	活用型-3	原形-3
1	し	動詞	自立	サ変・スル	連用形	する
2	さ	動詞	自立	サ変・スル	未然レル接続	する

	表層形-4	品詞-4	細分類1-4	活用形-4	活用型-4	原形-4
1	、	記号	読点	*	*	、
2	せ	動詞	接尾	一段	未然形	せる

表２

出力されたCSVファイルはMicrosoft Excelなどのアプリケーションソフトウェアで処理することができる。例えば、各形態素の表層形、活用型、原形などをキーに集計を行えば、先の検索結果381例の中で、使役形と非使役形がそれぞれどのような割合で用いられているかを求められる。表3は、コーパス中で最も多く用いられていた10の「強制～＋サ変」表現を取り上げ、それぞれの使役形・非使役形における出現度数を示したものである。[26]

	主要部	総数	非使役形			使役形	比率
			する・し	され	計	させ	非使役形 / 使役形
1	移住	75	2	3	5	70	5 / 70
2	送還	45	5	40	45	0	45 / 0
3	連行	25	5	20	25	0	25 / 0
4	終了	17	11	1	12	5	12 / 5
5	収容	12	2	9	11	1	11 / 1
6	排除	12	8	1	9	3	9 / 3
7	退去	11	0	1	1	10	1 / 10
8	着陸	9	0	0	0	9	0 / 9
9	労働	8	0	0	0	8	0 / 8
10	解除	6	3	3	6	0	6 / 0

表３

この結果から次のことがわかる。表３における「強制～する」の使役形・非使役形での分布は、ほぼ小林 (2004) の指摘通りの形になっている。しかし、よく観察してみると、少なからず例外的な事例が存在している。それらの多くは規範から逸脱した表現であり、単なる記述のミスと見受けられるものも少なくない。それでも、そのような表現が実際になされているというこ

とは事実である。どのような場合に非標準的な表現が起こりやすいか、あるいは間違いが生じやすいかという問題も、言語研究の重要な一部分である。mconc.rbによる処理を行うことで、誤用を含む様々な実例と統計データをコーパスから抽出し、より現実的な観点から言語事象を分析することができる。

4.2.4 mconc.rbによるコーパス処理の注意点

mconc.rbによるテキスト処理は、言語研究にとって有用であると思われるが、計算機による機械的な処理である以上、いくつかの注意点や制限がある。ここではそれらについて簡単に触れておく。

第一に、mconc.rbは、対象ファイルに含まれる日本語テキストを句点（。）をキーにして文に分割した上で、形態素解析にかける。したがって、句点が文の区切りとして働いていない箇所では望みどおりの結果が得られない。例えば、次のようなリスト項目（記事のメインの文章から独立した、箇条書きになっている部分）で、このことが問題になる。[27]

> **#東京23区（約852万人） - 旧東京府管下にあった東京市35区の区域。→特別区、日本の市の人口順位**
>
> **（記事「東京」より）**

この中で、「#東京」から「区域。」までは１つの文として形態素解析の対象となる。一方、「→特別区」以降は、文字列終端に句点を持たないため文としてみなされず、形態素解析を受けない。しかし前者は本当に「文」だろうか。また、仮に文だとした場合、後者を文でないと断じることは本当に妥当だろうか。

Wikipediaの編集方針の中に、句点の使用について厳密な規定はなく、結果的にそれは、記事の性質や、執筆者のスタイルに大きく左右される部分となっている。また、単純な編集ミスが比較的生じやすい部分でもある。そのため、問題の絶対的な解決法はない。しかしmconc.rbでは、句点の使用に関する「揺れ」が生じやすいリスト項目のテキストを一括して読み飛ばすオプションを用意している。このオプションをオンにすれば、句点使用が比較的

安定している記事本文のテキストのみを分析の対象とすることができる。

句点の使用に関しては、もう一つ重要な問題がある。句点が文の終端ではなく途中で用いられている事例への対応である。これには、引用文の最後に句点が用いられている場合や、文の内容の一部として用いられている場合がある。例えば次のような箇所において、単純に句点で文字列を分割するのは誤りである。

著作権法第10条2項では、「事実の伝達にすぎない雑報及び時事の報道は、前項第1号に掲げる著作物に該当しない。」と定義されている。
（記事「著作権」より）

作品の途中で「。」を付けられるところを探してみるとよい。
（記事「句切れ」より）

そこでmconc.rbでは、句点をキーにした文の分割を行う際、文字列中の開括弧の有無と個数を確認し、対応する括弧によってそれらが閉じられるまで文の分割を行わない方式を採っている。

最後に、形態素解析システムMeCabによる解析精度が100%ではないということを述べておきたい。MeCabは、コーパスからの学習と外部辞書によって形態素解析を行っているため、新語や、語句の非標準的用法に対しては、必ずしも正確な判断ができない。このことは、Wikipediaのように、新しい項目に関する記事が次々に追加されていくような性質を持ったデータを扱う際には、重要な事実である。しかし、MeCabには高度な推測メカニズムが実装されており、未知語を含むテキストも、ある程度の精度で解析できる。多くの場合、十分実用に耐えうるレベルであると考えられる。

5．おわりに

本稿では、誰もが閲覧・執筆・編集を行えるインターネット上の百科事典Wikipedia日本語版をコーパスとして用いた言語研究の可能性について考察した。また、Wikipediaの記事テキストを処理するためのツールキットを開

発し、それを構成するwp2txt.rbとmconc.rbという２つのRubyスクリプトについて解説を行った。

本稿で示した手法の重要な点は、すべてオープンソースのデータとソフトウェアにより構成されているということである。Wikipediaのアーカイブデータと、それを分析するためのツールキットは、誰でも自由に利用することができる。これにより、研究者間で同一のデータや分析手法を共有することが可能になる。また、すでに公開された研究成果に対して追試・修正・拡張・応用を行うことも非常に容易になる。

とはいえ、Wikipediaはまだ新しいプロジェクトであり、今後、何らかの仕様変更が生じることが大いに考えられる。Wikipediaを利用した研究は、良くも悪くもその影響を受けることになるだろう。また、現段階でのツールキットの機能は必ずしも十分ではなく、操作系も洗練されているとは言い難い。今後も様々な方面からのフィードバックを得ながら、ツールキットの改良を含め、Wikipediaをコーパスとして用いた研究手法の開発を続けていく必要がある。[28]

注

1 形態論や統語論、あるいは文体論のように、「形式」に重きを置く研究分野だけでなく、現在は認知言語学のように「意味」に重点を置く研究分野においてもコーパスを用いた研究が進められている。(Stubbs 2002; Deignan 2006; Gries and Stefanowitsch 2006)

2 代表性を有する大規模日本語書き言葉コーパス (http://www.tokuteicorpus.jp/)

3 Wikipedia日本語版 (http://ja.wikipedia.org/)

4 Wikipedia, The Free Encyclopedia (http://www.wikipedia.org/)

5 Wikipedia: 日本語版の統計 (http://ja.wikipedia.org/wiki/WP:JWS)

6 GNU Free Documentation License (http://www.gnu.org/licenses/fdl.html)

7 The Free Software Foundation (http://www.fsf.org/)

8 Wikipedia: 削除の方針 (http://ja.wikipedia.org/wiki/WP:DEL)

9 Wikipedia: 基本方針とガイドライン (http://ja.wikipedia.org/wiki/WP:POL)

10 Wikipedia: データベースダウンロード (http://ja.wikipedia.org/wiki/WP:DD)

11 ここで言う「行」は、一般的な行を指すのではなく、計算機にとっての行、す

なわち「終端に改行コードを持った一連の文字列」を意味する。

12 ただし、現在開発中のWindows GUI版では、Rubyおよびbz2をあらかじめインストールしておく必要はない。

13 Rubyの詳細については、開発者自身による解説書である、まつもと・石塚 (1999) の他、包括的なリファレンスとして、Thomas, Fowler, and Hunt (2005) が参考になる。

14 本稿執筆時の最新版は１.８.５である。なお、ツールキットの開発はDebian GNU/Linuxで行い、Windows XP SP2上でテストを行った。Windows版のRubyにはいくつかのバージョンがあるが、テスト環境では、One-Click Ruby Installer for Windows (1.8.5) を用いた。

15 MeCab: Yet Another Part-of-Speech and Morphological Analyzer (http://mecab.sourceforge.jp/)

16 解析アルゴリズムの詳細については、工藤・山本・松本 (2004) を参照。

17 wp2txt.rbでは、このようなHTML文字実体参照の解決に加えて、Unicode数値文字参照の解決も行う。これにより、様々な言語に固有の文字を表現する数値コードが実際の文字に置き換えられる。ただし、どこまで正確に変換できるかは、Rubyの文字コード処理機能の実装に依存する。

18 正規表現 (Regular Expression) とは、文字列のパターンを論理的な形式で記述するための方法である。文字列検索のきわめて有用な仕組みであり、多くのプログラミング言語で用いられている。また、テキストエディタやワードプロセッサの検索機能に組み込まれていることも多い。詳しくはFriedl (2002) などを参照。

19 YAML.org (http://www.yaml.org/)

20 より正確には、文字列抽出条件の配列を値として含むハッシュのキーである。

21 より正確には、^と$はそれぞれ「行」の始点と終点を表す記号であるが、ここでは便宜上「文字列」の両端を表すものとして解説している。

22 したがって、形態素1個に対応するパターンとして15＋１通りの記述法が存在することになる。

23 ただし、設定ファイルとコマンドラインとで同じオプションが指定された場合、コマンドラインの方が優先される。

24 ここで、３番目の形態素のfeature要素に記述されている.*という正規表現は、何らかの文字が０個以上並ぶことを意味する。また、条件式の最後に不特定のパターンを加えたのは、動詞のヴォイスやアスペクトについての情報を得るためである。

25 スペースの都合により、この表ではいくつかの形態素情報を省いている。また、オリジナル文については、重要な箇所のみを記載している。

26 第10位の「解除」と同じ出現頻度値を示した語として「疎開」があったが、ここでは単純に、Excelでのソート順で上位となる「解除」の方を優先させた。

27 WikipediaのWiki記法において、先端に#を付加された行は、連番つきリスト項目としてみなされる。その他にも、リスト項目に関してはいくつかのWiki記法が存在する。

28 ツールキットの開発には、数々のオープンソース・ソフトウェアを利用させていただいた。特に、形態素解析システムMeCabの開発者である工藤拓氏と、プログラミング言語Rubyの開発者であるまつもとゆきひろ氏には、素晴らしいソフトウェアを無償で公開されていることに対し、心からの感謝を申し上げたい。

参考文献

Deignan, Alice. 2006. *Metaphor and Corpus Linguistics*. Amsterdam: John Benjamins.

Friedl, Jeffrey E. 2002. *Mastering Regular Expressions*, 2nd ed. Sebastopol, CA: O'Reilly and Associates. 田和勝訳.『詳説 正規表現 第２版』東京: オライリー・ジャパン.

Gries, Stefan Th. and Anatol Stefanowitsch. 2006. *Corpora in Cognitive Linguistics: Corpus-Based Approaches to Syntax and Lexis*. Berlin: Mouton de Gruyter.

小林英樹. 2004.『現代日本語の漢語動名詞の研究』東京: ひつじ書房.

工藤拓・山本薫・松本裕治. 2004.「Conditional Random Fields を用いた日本語形態素解析」『情報処理学会研究報告』NL-161, 89-96.

まつもとゆきひろ・石塚圭樹. 1999.『オブジェクト指向スクリプト言語Ruby』東京: アスキー出版局.

斉藤俊雄・赤野一郎・中村純作編. 2005.『英語コーパス言語学―基礎と実践― 改訂新版』東京: 研究社.

Stubbs, Michael. 2002. *Words and Phrases: Corpus Studies of Lexical Semantics*. Oxford: Blackwell. 南出康世・石川慎一郎監訳.『コーパス語彙意味論―語から句へ―』東京: 研究社.

鷹家秀史・須賀広. 1998.『実践コーパス言語学―英語教師のインターネット活用』東京: 桐原ユニ.

Thomas, Dave, Chad Fowler, and Andy Hunt. 2005. *Programming Ruby: The Pragmatic Programmer's Guide*, 2nd ed. Raleigh, NC: Pragmatic Bookshelf. まつもとゆきひろ監訳. 田和勝訳.『プログラミングRuby 第2版 ライブラリ編／言語編』東京: オーム社.

Method for Using Wikipedia as Japanese Corpus

Yoichiro HASEBE

Key words: corpus linguistics, Japanese, Wikipedia

Linguistic research and its methods using large-scale corpora have been attracting more and more attention in recent years. Major projects of constructing large-scale corpora are now being carried out not only for English, of which there are several major corpora such as the British National Corpus, but also for many other languages. At present, however, there are few corpora of written Japanese widely available for researchers. One of the reasons why a large corpus is difficult to come by is that numerous procedures must be completed before the copyright issues are cleared. It is not a matter of just collecting a large amount of text and sharing it among researchers.

There is, however, one source where a great deal of Japanese text is continually submitted and accumulated in a form that is completely open to the public. That source is Wikipedia. Although some restrictions do apply, as is the case with any other medium, Wikipedia offers quite a large set of linguistic data that reflects the present state of both the grammar and the vocabulary of the Japanese language. This is favorable for many linguistic approaches in a synchronic perspective. Moreover, since the compressed package of all the articles is published regularly for archiving purposes, it is also hoped that researchers will use these data to investigate the semi-diachronic phenomenon as well.

With the above facts as background, this paper suggests a method to utilize Wikipedia in linguistic researches based on corpora of written

Japanese. A computational toolkit to effectively access and analyze the text data in the archived file is presented. This toolkit comprises two programs written in the programming language Ruby. One, called wp2txt.rb, extracts article texts from the original XML data, and converts the bare text with Wiki and HTML tags sprinkled everywhere into plain text suitable for analysis. The other program, called mconc.rb, matches up a morphological collocation pattern (described using Regular Expressions in a configuration file) in the Wikipedia article texts. It also outputs results in CSV format so that it is able to process the data on ordinary spreadsheet application software.

By using the archived data of Wikipedia and the toolkit introduced in this paper, researchers can easily retrieve examples and statistical data of a particular linguistic phenomenon. Moreover, since the Wikipedia data and the toolkit are available as open-source, the procedure of particular research and its resulting data can be tested, refined, or expanded, by other researchers, making it possible for communities to build an effective inter-research network.