言語リソースとしてのWikipedia

3.1 はじめに

世界最大のオンライン百科事典であるWikipediaを活用しておられる方々は多いのではないかと思います。信頼性の問題が指摘されてはいるものの質・量共に史上最大の百科事典から得られる情報量の多さ故に、筆者も調べ物の手段として日常的に活用しています。また、筆者の場合はソーシャルデータとしてのWikipediaの研究を手がけている関係でWikipediaのPageview Countに大いに関心を持っているところです。

このようにWikipediaを研究対象として捉えている研究者はたくさんいらっしゃるようです。例えばDBPedia[*1]やその日本語版[*2]はWikipediaの情報を抽出し、LOD(Linked Open Data)に変換して公開する試みでして、セマンティック・ウェブといった研究などに活用されているそうです。あるいは自然言語処理の研究分野でもWikipediaは膨大な文例を収蔵する言語リソースとして認識されており、様々な活用方法が提案されています。そこで本稿では、筆者の研究からは離れて、言語リソースとしてのWikipediaをテーマに紹介します。

3.2 「Unix考古学」の執筆

筆者が言語リソースとしてのWikipediaに関心を持つのには少々訳がありまして、実は今年の4月に「Unix考古学[*3]」という本を出版しました。UNIXに関わる歴史的な事実を文献を辿りながら紹介していくこの本は、強いて言えば歴史書にカテゴライズされると思います。

同書はもともと、月刊誌「UNIX USER」に2003年から2005年にわたって掲載された26編の記事を整理・再編集して単行本にまとめたものなのですが、取材や図書館通いをすることなく、Googleの検索エンジンで集めた文献だけで執筆された書籍であるところにも特徴があります。今日ならいざ知らず、1998年にGoogleの検索エンジンが公開されてから数年しか経っていない当時としては、これはかなり無謀な執筆スタイルでした。

もっとも、当時の筆者には(今考えれば根拠が極めて薄弱な)確信がありました。それは1980年代に現れた新しい職業の中に「データベース検索技術者」、通称「サーチャー」という職業がありました。それを紹介するテレビ番組のなかで、小説家とサーチャーがタッグを組んで新作を執筆するという事例が語られたのですが、そこではサーチャーが小説家に対し「あなたは執筆するための資料集めを一切する必要はありません。あなたが必要だと思う情報を教えてくれれば、私がすべてデータベースの中から引き出してみせます」と豪語したエピソードが紹介されていました。まだ就職したての新人だった、当時の私は「そんなのことができるのか?」と非常に懐疑的な印象を持ったのでした。それから十数年経過して、連載の執筆依頼をもらった際、このエピソードを思い出した筆者は「Googleの検索エンジンが使える今なら、私にもこのアプローチが可能かもしれない」と考えたのでありました。

実は、執筆依頼を受諾してから最初の記事が発表されるまで、半年程度の猶予がありました。編集部としてはこの期間を使って数回分の記事を書きためることを期待していたのですけども、筆者はほとんどの時間をGoogleを使って欲しい文献を探し当てるキーワードの選び方と並べ方の順序、つまり検索パターンを発見するために費やしました。結果、半年後に完成していたのは第1回と第2回のみで、後々、厳しいツケを払い続けることになりました。このようにして検索エンジンを片手にトピックを「どこまでも掘り下げて」文章を書き加えていく、ちょっと風変わりな執筆スタイルに定着しました。

3.3 Drilldown Search

今では「どこまでも掘り下げて」を表現するのにピッタリなDrilldownという用語があります。Wikipediaの"Drill down"[*4]のページによれば「何かに注目しながら詳細なデータを得るために、ある場所から別の場所へと移動することを意味する」と定義されています。筆者のケースはこのページで例示されているオン

*1 DBPedia(http://wiki.dbpedia.org/)。
*2 DBPedia日本語版(http://ja.dbpedia.org/)。
*3 Unix考古学(http://asciidwango.jp/post/142281038535/unix%E8%80%83%E5%8F%A4%E5%AD%A6-truth-of-the-legend)。
*4 Drill down(https://en.wikipedia.org/wiki/Drill_down)。

ラインユーザやウェブサーファーの事例が最も近いのだろうと想像していますが、注目するポイントが「計算機の歴史に関わる、あまり認知されていない文書」を発掘することにあるので、作業の中身は非常に特殊なケースなのではないかと考えています。

例えば・・・

> 7th Edition Unixではカーネルも全面的にC言語で書き換えられた。これは一般にも割と有名な話。この作業にあたったのはKen ThompsonとDennis RitchieとSteve Johnsonの3人であること、またSteve JohnsonはPortable C Compiler(PCC)の開発者であったことからこの作業に加わったことはDennis Ritchieが残した複数の文献(論文、講演資料、インタビュー)から確認することはできる。では、Steve Johnson自身が一連の開発作業について言及した文献はないか?

あるいは・・・

> BSD UNIXの開発において、4BSDをリリースした際、主に3BSDとの比較でパフォーマンスが低下したことを指摘する批判が出て、その対応のためにUCBのCSRGはパフォーマンスチューニングを施した4.1BSDをリリースした。ここまではUNIXの歴史を語る多くの文献で言及されている。しかし、その実態はDavid Kashtanという人物の非常に攻撃的な批判記事にBill Joyが激怒し反撃を行った事実があることがKirk McKusickの文書で明らかになっている。それではKashtanの批判記事、及びそれに対するBill Joyの反論の内容が分かる文献はないか?

・・・といった歴史的な事実に関するDrilldownを試みるとなると、検索は非常に厄介な作業になってきます。探し当てた文献の関連箇所を読み込み、その中から新たな文献を探索するキーワードを拾い出す一連の検索サイクルを繰り返して、毎月6〜10ページの記事を書くことは非常にエネルギーと集中力を要する作業で、小説家の執筆というよりはジャーナリストの記事作成に近い仕事なのではないかと想像しています。

事実、連載終了後から12年が経過して、単行本としての出版が決まった際、相当の分量の書き直しや新たな書き下ろしが必要になったので、12年前を思い出して同様の作業を再度試みたのですが・・・歴史的な事実のDrilldownは、年老いた今の筆者にはとても耐えられない程のハードな作業に変化していました。「この作業の一部だけでもコンピュータにやらせられないだろうか?」ふと、そんなことを考えるようになっていきました。

3.4 固有表現認識(NER)

以来、筆者は「歴史的な事実のDrilldownの機械化」を考え続けています。これは当初、研究というよりは執筆活動のための道具立てとしての側面が強かったのです。もちろん「これさえあれば執筆依頼を受けるハードルが下げられる」との少々ヨコシマな考えもありました(笑)。

機械化、つまり筆者の文献探索作業を模倣するソフトウェアを作るためには、まず「どのようにして文献を探していたのか?」を明らかにする必要があります。前述の記事執筆の経験から体得した「検索エンジンによる文献探索と事実情報の抽出」で最も重要だったノウハウは、固有名詞に着目することでした。在り来たりに思えるでしょうが人名、組織名、それから計算機の歴史的事実の場合は過去のコンピュータの型番や愛称などが該当します。これらの固有名詞をできるだけ拾い集めその組み合わせを変えて検索キーワードを指定すると、検索結果を望む方向に大きく絞れることが多かったのです。そこで英語テキストから固有名詞を抽出する技術を探すところから手を付けることにしました。

当然のことながら自然言語処理の研究領域の話になるのですが、調べ始めてすぐに気づいたことが2つありました。

(1) 現在は統計的自然言語処理が主流
(2) 日本の自然言語処理研究と英語圏の自然言語処理研究では研究の様相がかなり違う

まず(1)に関しては、現在の筆者とは研究の目的もゴールもまったく違うのですが、いずれも統計学に基づく技術をベースに研究が進められていることです。特に、筆者の場合はデータ分析の基盤技術も研究の対象となっているので、基盤技術だけを見れば多くの共通点が見つけられるように感じました。

次に(2)については、これまで自然言語処理と言うと暗黙のうちに「日本語の」と仮定していたところがあり、形態素解析や機

械翻訳といった日本語に関わり深い研究が主要なテーマだと筆者は理解していたのですが、英語の自然言語処理研究では（当然のことながら）取り組まれているテーマがだいぶん違うとの印象を受けました。

筆者が欲していた「英語テキストから固有名詞を抽出する技術」も固有表現認識（NER：Named Entity Recognition）と呼ばれる英語での自然言語処理研究では主要な研究テーマの1つになっているそうです。

3.5 Natural Language Toolkit（NLTK）で利用できるNER

今日ではNatural Language Toolkit（NLTK）にNERが搭載されているので、図-1のようなPythonスクリプトを使って比較的お手軽に固有表現の抽出を試すことができます。

このスクリプトは英文のテキストファイル名を渡すとNERを実行して図-2のような出力をします。

出力は各行単語、品詞、NEタグが表示されています。NEタグの先頭がB-の場合は固有表現の先頭、I-の場合は継続語を意味します。PERSONは人名、ORGANIZASIONは組織名、GSPは"Geo-Socio-Political group"の略です。"Hillary Clinton"や"Donald Trump"が人名として認識されていることが分かります。

次に、もう少し大きなテキストでの事例として「Unix考古学」の執筆でもお世話になったMarshall Kirk McKusickの"Twenty Years of Berkeley Unix"のテキストから人名を抽出してみました（図-3）。

誤認識もありますが、Ken ThompsonやDennis Ritchie、Bill Joyと言った有名人だけでなく、Kirk Mckusick以外のBSD UNIXの開発メンバーであったOzalp Babaoglu、Sam Leffler、Mike Karels、Keith Bosticの名前も正しく認識されています。このように、特定の目的（例えばUNIXの歴史に関するトピック）のために特別なチューニングをしなくても、それなりの精度で固有表現を抽出できるのが、最近の統計的自然言語処理の研究成果なのでしょう。

3.6 統計的自然言語処理と"コーパス"

さて「特別なチューニングなし」に「それなりの精度で抽出できる」不思議さを知るには統計的自然言語処理の研究を勉強するしかないのでしょうが、解説書を読むと頻繁に登場するのが"コーパス"という用語です。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import nltk
import sys

if __name__ == "__main__":

    param = sys.argv

    args = param[1]

    with open(args, 'r') as f:
        sample = f.read()

    sentences = nltk.sent_tokenize(sample)
    tokenized_sentences = [nltk.word_tokenize(sentence) for sentence in sentenc\es]
    tagged_sentences = [nltk.pos_tag(sentence) for sentence in tokenized_senten\ces]

# for NLTK 2.7
#   chunked_sentences = nltk.batch_ne_chunk(tagged_sentences)

# for NLTK 3.0
    chunked_sentences = nltk.ne_chunk_sents(tagged_sentences)

    entity_names = []
    for tree in chunked_sentences:
        print nltk.chunk.tree2conllstr(tree)
```

図-1 named_entity_recognition.py

Wikipedia英語版によると、"コーパス"とは言語学において使用される「巨大で構造化されたテキストのセット」で「統計的な分析や仮説検定、頻度のチェック、特定の言語のルールに基づく正当性の確認に使われる」そうです。筆者のような門外漢にはこの説明だけではピンと来ないのですけども、一般に広く知られているジップの法則(Zipf's law)「出現頻度がk番目の単語が全体に占める割合は1/kに比例する」という経験則を思い出してもらうと「人手が書いた文章と統計学の不思議な関係」を少しイメージしやすくなるのではないかと思います。つまり「人間は作文において無意識のうちに統計学的に偏りのある振る舞いをしている」。それが統計的自然言語処理がうまく機能する理由なんだろうと想像しています。

実は、この統計学的に偏りのある振る舞いを、筆者は「Unix考古学」の執筆経験を通して直感的に体験していました。例えば、Dennis Ritchieはホームページ[*5]でUNIXの開発経緯に関して述べた多数の文書を公開しています。執筆の際には大変お世話になったのですが、複数の文章をなんども読み直すうちに「あれ、この言い回しは他のどこかでも見たことがある・・・」と気

```
$ cat NHK-short.txt
A US poll shows that Democratic presidential nominee Hillary Clinton's lead over her Republican rival, Donald Trump, has shrunk to 3 percentage points.
$ ./named_entity_recognition.py NHK-short.txt
A DT O
US NNP B-GSP
poll NN O
shows VBZ O
that IN O
Democratic JJ B-ORGANIZATION
presidential JJ O
nominee NN O
Hillary NNP B-PERSON
Clinton NNP I-PERSON
's POS O
lead NN O
over IN O
her PRP$ O
Republican JJ B-ORGANIZATION
rival NN O
, , O
Donald NNP B-PERSON
Trump NNP I-PERSON
, , O
has VBZ O
shrunk NN O
to TO O
3 CD O
percentage NN O
points NNS O
. . O
$
```

図-2 サンプルの英小文とNLTKの固有表現認識の実行結果

```
Alan Nemeth
Babaoglu
Beranek
Berkeley
Berkeley Software Design
Berkeley Software Distribution
Berkeley Unix
Bert Halstead
Bill Jolitz
Bill Joy
Bob Baker
Bob Fabry
Bob Guffy
Bob Kridle
Bostic
Casey Leedom
Chuck Haley
DARPA
Dan Lynch
David
Dennis Ritchie
Dickinson R. Debevoise
District Judge
Domenico Ferrari
Duane Adams
Eugene Wong
Fabry
Fateman
Ferrari
Freely Redistributable Marshall Kirk McKusick Early History Ken Thompson
Haley
Hibler
Jeff Schriebman
Jerry Popek
Jim Kulp
John Reiser
Jolitz
Joy
Karels
Keith
Keith Bostic
Keith Lantz
Keith Standiford
Ken Thompson
Laura Tong
Leffler
Linux
Lucasfilm
Math
Michael Stonebraker
Mike
Mike Karels
Mike Muuse
Networking Release
Ozalp Babaoglu
Pascal
Pauline
Peter Kessler
Professor Domenico Ferrari
Professor Richard Fateman
Professors Michael Stonebraker
Ray Noorda
Rick Macklem
Rick Rashid
Rob Gurwitz
Robert Elz
Sam Leffler
Schriebman
Schwartz
Statistics
Support Meanwhile
Susan Graham
System
System III
System Manual
System V
Tahoe
Thompson
Tom London
Unix
Unix Early
Utah
```

図-3 PERSON of "Twenty Years of Berkeley Unix"

*5 Dennis Ritchie(https://www.bell-labs.com/usr/dmr/www/)。

づくことが度々ありました。つまり、読者には「執筆者の癖」と理解できる記述が、統計的なテキスト分析では「統計学的な偏り」として把握できるようです。

ちなみにDennis Ritchieの最も分かりやすい癖は、PCCの開発者をSteve Johnsonと記していたところです。Ritchieの記述にはすべてこの名前が使われていたのですが、実は彼の本名はStephen Johnsonで、彼自身が執筆した論文やWikipediaページなどではStephenと記述されてます。Ritchieの誤解だったのか、それとも彼はベル研ではSteveという愛称で呼ばれていたのか定かではありませんが、いずれにせよ、事実の後追いをしている筆者には「どんな条件で検索しても文献がまったく引っかからない」非常に困惑させられる経験をすることになりました。

どうやら統計的自然言語処理とは人間が書いたテキストを対象に統計学的な偏りから様々な新たな知見を得る研究のようです。

3.7 WikipeidaからNEコーパスを生成する

NLTKの話に戻ります。NLTKのNERの概要は"入門 自然言語処理"（原書："Natural Language Processing with Python"）の第7章「テキストからの情報抽出」で解説されていますが、その実装はNLTK Corpora[*6]の1番目にリストされている"ACE Named Entity Chunker（Maximum entropy）"です。

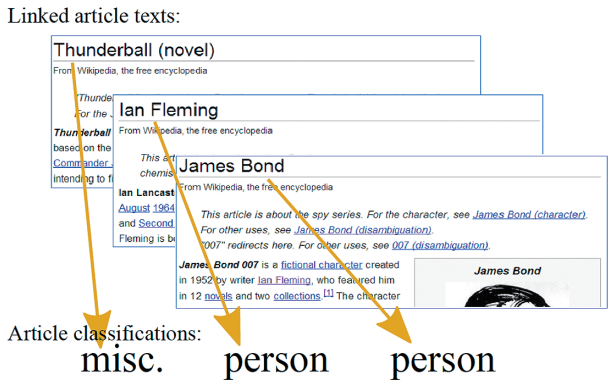

図-4 Deriving training sentences from Wikipedia text

この「最大エントロピー」と呼ばれる機械学習モデル（同書の「6.6 最大エントロピー分類器」に解説があります）は、固有表現タグを注釈したNEコーパスを使用します。単語に「人名」や「組織名」といった固有表現タグを付与するNEコーパスの作成は人間が介在せざるを得ない非常に手間のかかる作業で、故に無料配布されているNEコーパスはほとんどないそうです。実際、NLTKの場合もAutomatic Content Extraction（ACE）が作成したコーパスを使って学習したチャンカーのみをpickleファイルとして配布しています。コーパスそのものは含まれていません。

このNEコーパスをWikipediaのページデータを使って機械的に生成しようという試みがあります。論文"Transforming Wikipedia into Named Entity Training Data[*7]"では、固有表現の注釈付きのコーパスを作成するために、Wikipediaの利用を提案しています。

Wikipediaの記事の中で記載された用語や名前は、多くの場合、適切な他の記事にリンクされています。そのような記事間のリンクを固有表現の注釈に変換することがこの論文の基本的なアイデアです。例えば「イアン・フレミング」の小説「サンダーボール」において「ジェームズ・ボンド」というキャラクターについて紹介する文章は、各々の固有表現について相互にリンクが張られていることが期待できます。「イアン・フレミング」の記事は、その表現が人物であることを示してくれるでしょうし、「サンダーボール作戦」の記事から、それが小説であることが分かります。このようにリンクを辿ると元の文章には自動的に注釈を付けることができるわけです（図-4）。

論文ではWikipediaからNERをトレーニングするための何百万もの文章を抽出して、巨大なコーパスを形成することができると述べています。そのプロセスとして次の4つのステップを掲げています。

1. すべての記事をエンティティクラスに分類する
2. Wikipediaの記事から文章を取り出す
3. リンクターゲットに応じて固有表現をラベル付けする
4. コーパスに収録すべき文章を選択する

*6 NLTK Corpora（http://www.nltk.org/nltk_data/）。
*7 Transforming Wikipedia into Named Entity Training Data（https://www.aclweb.org/anthology/U/U08/U08-1016.pdf）。

この手順を使って、論文では標準的なCONLLカテゴリのエンティティクラス(LOC、ORG、PER、MISC)に基づくNEコーパスの生成を試みています。

現在では英語版は500万以上、日本語版でも100万を超える記事を収蔵するWikipediaを対象にこの手順を実行するのは明らかにビッグデータ処理になります。特に論文が記事の分類のために示すブートストラッピング、すなわち一部は人手でクラスを判別する方法はコーパスを完全に機械的に生成する上での困難な課題でしょう。

3.8 おわりに

図らずも、筆者が考える「歴史的な事実のDrilldown文献探索」の機械化は、最新の自然言語処理研究の成果を使えばかなり前進するのではないか？という予感と、筆者が研究として取り組んでいるWikipedia分析のために開発した基盤技術がこの取り組みにも活用できる可能性があることに少し驚きを感じているところです。

文献から人名やシステム名を抽出する仕事はNERを使えば実現可能ですが、その認識精度を上げるためにはNEコーパスの入手とそれを使った学習が欠かせません。独自のNEコーパスを生成するためのリソースとして、日頃から慣れ親しんでいるWikipediaが活用できるというのは朗報でした。残る課題は500万ページ以上ある英語版Wikipediaから人物のページを拾い上げること・・・。

この問題は、筆者が手がける歴史書の執筆などではあまり問題にはならないように思います。歴史書の執筆とは筆者が定めたテーマ、すなわち時代や人物、事件について深く掘り下げていく行為にほかならないからです。英語版Wikipediaから執筆テーマに関連する人物のページを収集する作業などは日常茶飯事です(笑)。ノンフィクションや報道記事を手がける記者や社会科学の研究者は、人物に限らず固有表現やその分類に精通しています。彼らの知識を集合知として集め共有する方法を考えるのが、問題解決の早道だと思います。

以上、筆者は思いもよらぬことから今日の自然言語処理研究を学べる機会を得ましたが、この研究成果は、現在研究で手がけているソーシャルデータの分析にも活用できるのではないかと考えている次第です。例えば「要因分析」。ソーシャルデータの時系列分析を行うと急激な変動、いわゆるバーストを検知することができますが、このバーストを引き起こした原因がなんであったかを調べるためにはバースト発生時の社会的動向を追跡する必要があります。筆者は検索エンジンなどを使って報道記事を集めることを考えていたのですが、そのための検索条件を決める手段として本稿で紹介した方法が使えるのではないかと考えてます。

執筆者:
藤田 昭人(ふじた あきと)
株式会社IIJイノベーションインスティテュート(IIJ-II)企画開発センター チーフアーキテクト。2008年IIJ入社。
構造化オーバーレイ研究の知見を活用したクラウドコンピューティング技術の研究開発に従事している。