検索エンジンに基づく多言語用例指南ツール:KIWI

山本真人† 田中久美子† 中川裕志‡
†東京大学 大学院 情報学環 ‡東京大学 情報基盤センター
E-mail: kumiko@ipl.t.u-tokyo.ac.jp, {masato-y,nakagawa}@dl.itc.u-tokyo.ac.jp

本稿では、ユーザーが入力した文字列に関する結果を検索エンジンから得て、用例を集計するツールについて報告する。ツールは言語依存の処理を一切行っておらず、web上にデータがあればどの言語でも用例を調べることができる特徴を持っている。このツールは、語学の学習者が、外国語での言い回しを調べる際に有用である。システムの概要を述べた上で、300件ほどの英熟語をツールを用いて調べた結果について報告する。

1 はじめに

インターネットの普及により、国際語としての英語へのニーズが高まると同時に、英語以外の言語に接する機会も増えている。このように外国語が身近な存在となった現在、生きた言語の用例を調べる機会が急激に増えてきている。

言語の用例を調べるには、古くから辞書が用いられてきた。辞書には精選された項目が記載されており、普遍的な用例を調べる用途には有用である。しかし、一方で今日的な用例が見つからなかったり、自分の望む具体例が載っていないことが多く、外国語の運用上は自分の語用が正しいのかどうか、不安が残ることも多い。

辞書にも載っていない用例を調べるとき役に立つ方法として、検索エンジンを用いることが挙げられる。自分の調べたい用語の一部を対訳辞書を用いて外国語に翻訳し、その用例を検索エンジンで調べるのである。たとえば、「無線LAN」をフランス語で調べる場合は、「無線」に相当するフランス語「sans fil」を入力する。すると最初の２０件を見るだけで「le reseau sans fil」「l'access sans fil」などと挙がる。

実は、さらに数百件くらいを眺めると、「l'internet sans fil」「les reseaux sans fil」などがより一般的に用いられるようであることがわかってくる。つまり検索エンジンの結果を集計すると、より確実な言語の使用例を知ることができるのである。ここで問題となるのは、用例を知りたい時に毎回500件や1000件の検索結果を走査するのが、人間には困難である点である。そこで、集計を自動化するツールが求められる。

この問題意識に基づいたツールについてはすでに前例があり、たとえば、Webcorp[5]やGoogleDuel[2] GoogleFight[4] などが挙げられる。しかし言語が英語だけに限定されていたり、あるいは２つの異なる入力語の検索件数の比較だけに限定されていたりと制約が多い。そこで、我々は多言語の用例を比較的自由に調べることのできるkiwiを作成している[7]。本稿では、その構成について簡単に紹介した後、AltaVistaを用いて前回の報告よりも大規模な実験を行った結果を示し、kiwiの有用性について論じる。

2 kiwiシステムの概要

kiwiはJava言語で書かれたシステムであり、ネットワーク上の検索エンジンを利用することが前提となっている。このため、高速に通信が可能なネットワークにつながった状態で起動されるソフトウエアである[7]。入力質問は現在では、

- ワイルドカード:
  例 “the pen is mightier than * ”
  “ * 爾番丹”
- 複数の文字列の比較 :
  例 “(les / le) jardin ”

の二つを実装している。システムは入力を検索エンジンに送り、検索に必要な検索結果をユーザの指定した量だけ得る。その上で、検索結果から相当部分を切り出し、整列して表示する。

具体的なkiwiの入出力の例を表1に挙げる。表の第3列には検索エンジンで上位に表示される出力を順に記載する。これらは用例として必ずしも適当であるとは言えない一方で、第2列のkiwiの出力ではもっともなものが列挙されている。

表 1: kiwi の入出力例と AltaVista の上位に挙がる用例

入力質問	1位	2位	3位	AltaVista の上位例			
*phone	cell	mobile	cellular	tele new	reverse j-	cell converts	best prepaid
小泉*	内閣	首相	純一郎	産業 の会	成器 親司	内閣 製麻	聡 としあき
*大統領	ブッシュ	クリントン	金大中	米 次期	ラーメン アメリカ	摂津の 元	ブッシュ フルーツ

kiwi の本質とは検索エンジンからコーパスを得た後の用例の処理、すなわち候補の集計の方法にある。これについては、さまざまな研究の方向性があるが、我々は言語に依存する処理を用いないという方法を貫くことにした。というのも、本ツールは辞書やデータがそろっていない言語において、より必要とされる可能性が高いからである。以上から、言語に依存しない用例処理について次節で説明する。

3 用例の処理

用例の処理は、具体的には、候補の切り出しと整列である。

ワイルドカードが前方文字列の検索「*X」、あるいは後方文字列の検索「X*」と用いられた場合、候補を切り出す必要がある。前方と後方では問題が対称であるため、後方に限って本節では論じる。

候補の切り出しは、用語抽出や頻出する n-gram の抽出と類似している。しかし、本稿での問題は、候補を切り出すデータとなる検索結果は数千〜万単語程度の小さなコーパスであり、また、動的に候補を得るため高速な処理が必要であるという 2 点に留意する必要がある。

ある文字列が候補かどうかは、直感的には

1. 頻出する
2. 適当な長さである (極端に短くも、極端に長くもない)
3. 後続する文字の種類が多い

という性質を満たすことで判断される。

1は、その文字列の実際の場での使用についての性質である。つまり Web という広大な空間での使用の実態を示す。したがって、正しい言語使用が多数派であるという前提をおけば、正解用例の指標となる。2は、長さに関する基準である。文法的な係り受け、動詞の格情報や選択制限などは多くの場合、適当な長さの文字列中での制約として出現する。このような言語使用が用例検索の対象であるなら、これも正解用例を示す指標となりうる。

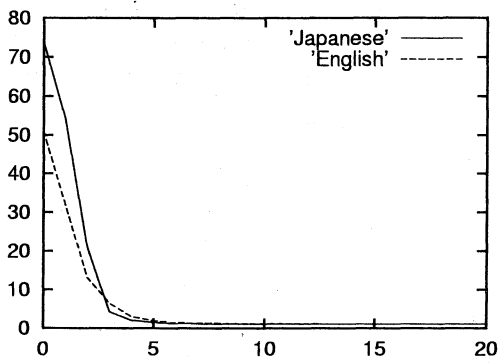

図 1: 単語の文字オフセットと分岐数

これに対して 3 は、Web という広大な空間で、別々の単語や言い回しを区別する文字列中の位置を示唆する指標である。例えば al という文字列の頻度が高くとも、大多数の場合に all の一部として出現するなら、むしろ all を取り出したい。これは all が単語であれば、その後には空白を挟んで多種の単語が現れ、al は単語でないので、後続する文字種は少ないということで理解できるだろう。

以上から、候補の切り出しと整列とを分けて考えることにした。順に以下で説明する。

3.1 候補の切り出し

「後続する文字種類数が多い」という定性的な性質をどのように判定するかは難しい。後続する文字種類数に着目して単語を切り出す方法は、用語抽出の分野で [1][3] において、単語を切り出しの最小単位としてさまざまな実験が行われている。我々の問題設定では、本研究が言語非依存のシステムを目指していることから、文字単位でこの考え方を応用する。

ここで、我々は「単語中の後続分岐数は減少する」

という性質に注目することにした。図1は、WSJおよび毎日新聞30Mから得た単語約9万単語をTrie構造化した際の、単語内での文字オフセットと、後続文字種類の分岐数との関係を示したものである。これを見ると、後続文字種類数は前から後へと一気に減少する。

そこで、Xを文字列、X_iをその頭のi文字、C_iをX_iの後続文字種類数とする。そして、

$$C_i > C_{i-1}. \tag{1}$$

を満たしたときに、X_iを候補として切り出すものとした。

この切り出し方法を用いると、候補に後続する文字種類数が小さい場合には、候補を切り出すことができない恐れがある。特に、検索エンジンの結果中に候補が1回しか現れない場合は絶対に切り出すことができない。しかし、用例は、多頻出であるという性質も持つため、このようなことはあまり起こらないと予想し、この手法を試すことにした。

また、この方法は言語に依存する処理は一つも用いてはいない。確かに、例えばスペースを特別扱いすることにより、欧米諸語の単語の切り出しの精度を向上させられる可能性がある。しかし、本稿の範囲ではすべての文字を等しく扱ってどの程度の性能が得られるかを検証するものとした。

3.2 候補の整列

候補の整列には、頻度と候補の長さを用いることはすでに述べた。特に、短い文字列は長い文字列に含まれてしまい、頻度が短い文字列に対して有利になってしまう点を補正する必要がある。そこで、我々は候補の評価関数として、

$$F(X) = freq(X)log(|X| + 1) \tag{2}$$

を用いるものとした。$freq$は頻度、$|X|$はXの長さを表す。もちろん、logを用いるかどうかなど、他に考えられるが、現状では、直感的に考えられるものの範囲で上記が実験的に良い結果を示したため、これを選択した。尚、本手法も言語に非依存の方法である。

4 評価

4.1 手順

本稿では英語の熟語を題材にkiwiの有用性を論じる。英熟語を選択した理由は、web上で多用されることが多いため、安定した量のデータを検索エンジンからダウンロードすることが可能であるからである。

表2: 英熟語の検索性能

		1位率	10位率	候補率
kiwi	前	77.0% ①	93.3%	97.0% ②
kiwi	後	78.5% ①	92.8%	96.3% ②
baseline	前	32.8%	75.5%	98.9% ③
baseline	後	33.6%	80.8%	97.4% ③

実験はつぎの手順で行った。

1. 熟語集[6]から3単語以上の熟語を計600件をランダムに80、用例が曖昧な熟語、例えば "* up with" に対する "come" と "keep" などは含まれていない。
2. 各熟語において、前方、もしくは後方の数語をワイルドカードで置き換え、各300件づつの入力質問を用意する。以下では置き換えられた数語を「正解」と呼ぶものとする。
3. この入力質問に合致する部分を検索エンジンで1000例ダウンロードし、kiwiを用いて集計する。
4. 前、後各300件につき、正解がkiwiの候補中に存在するかどうかを調べ、次の評価値を算出する。
   1位率 候補の第1位に正解が挙がった割合
   10位率 候補10位内に正解が挙がった割合
   候補率 候補に正解が上がった割合

この他、検索エンジンに質問を入力し、(kiwiでは集計していない) 生の結果の第1位、上から10位内、全体を見て、候補が存在するかどうかも調べ、これをベースラインの1位率、10位率、候補率とする。

4.2 用例検索性能

結果を、表2に示す。2,3行目がkiwiの結果、4,5行目はベースラインの結果である。ベースラインに比べ、1位率、10位率共にkiwiがより高い精度を示している。特に、1位率に関しては、ベースラインが30%強程度であるのに対し、kiwiでは倍以上の80%近い精度を示している。また、10位率に関しても検索エンジンでは80%程度であるが、 kiwi用いることで90% 以上の精度を得ることができる。

これによりkiwiの集計処理の有用性が分かる。

kiwiを用いても、正解を得ることができない場合の原因としては、以下に述べる3つの要因が考えられる。表2の評価値は、これらの要因が混ざり合った結果であるため、以下で各項目に分けて分析する。

検索エンジンの問題

検索エンジンの結果に正解が1回も含まれないと、正解は挙がらない。これは、kiwiを原因として起きる問題ではない。このようなことが起きる度合いは、(100 - 表2の③欄)% の値として示される。

逆に、③欄の値は、kiwiの上限値を与える。②の値がこの上限に近いことは、kiwiのようなツールの潜在的な有用性が高いことを示唆している。

正解の問題

想定している正解がweb上に頻出する語用ではない場合、本稿の実験では不正解となる。たとえば、入力質問 "be anxious *" で "for" を正解としていたところ、"to" が1位に挙がる場合も不正解である。これは、kiwiを原因とする不正解というよりも、正解の用意の仕方の問題である。

本研究の検証においては用例の観点からの真の正解の定義が難しい。一般に、語学の学習者が検索エンジンを用いて用例を調べる場合には、入力質問に工夫をする必要があることは暗黙の了解事項である。具体的には、検索エンジンになるべく頻出しそうで、かつ一般的すぎる語は避けるといった工夫である。このような条件を満たす入力質問を持つ正解の集合を用意して実験することが本来は望ましいが、すると機械的な入力質問の構築は難しい。

そこで、正解の問題を含んだまま実験し、可能な範囲で要因を分析する。表2中では、正解の問題は、

(③ - ②)+(② - ①) %

中に含まれる。各項を以下で分析する。

kiwiの問題

上記の式の第一項は切り出しの限界に正解の問題を、第二項は整列能力の限界に正解の問題を混在させた数値となる。切り出しについては、原理的に不可能な場合があり (§3.1参照)、検索エンジンの結果中の正解頻度が低い場合、特に頻度が1であると切り出せない。同様に、整列も頻度と長さを用いている以上、頻度が低くなると正解を上位に得ることができない。このように頻度が低くなってしまう原因には、正解の問題をも要因としており、必ずしもkiwiだけの問題ではない。

ただし、第一項の値は1～2%と小さい。つまり正解の問題を含んだとしても、切り出しにはほとんどの場合成功していることを逆に表しているといえる。一方で、整列の問題は正解の問題とあいまっているとはいえ、未だ改良の余地がある。今後は整列のためのよりより評価関数を考えることが重大な課題となろう。

5 結論

本稿では、検索エンジンを利用した用例指南システムkiwiについて報告した。kiwiはユーザが質問を入力すると、その語を検索エンジンに問い合わせ、検索結果を集計することにより、用例をユーザに提示する。

kiwiの特徴は集計に必要な言語の解析処理を言語に非依存のものにしている点にある。この性質により、システムは多言語に応用することができ、言語の解析技術や辞書が十分に整備されていない言語であっても、web上にデータさえあれば、用例を調べることができる。実際には、後続文字種類数に着目した方法を用いて候補の切り出しを実現し、頻度と候補の長さに応じて整列して提示する。

600件の英熟語を題材にkiwiの性能を調べた。その結果ツールの集計が機能していることが示され、また潜在的な有用性が高いこともわかった。今後は、複数の検索エンジンを用いた実験を行った後、ツールの公開を目指したい。

参考文献

[1] T.K. Frantzi and S. Ananiadou. Extracting nested collocations. *16th COLING*, pages 41–46, 1996.

[2] GoogleFight. Google fight : Make a fight with googlefight, 2000. Available at `http://www.googlefight.com/`.

[3] H. Nakagawa and T. Mori. A simple but powerful automatic termextraction method. In *Computerm2: 2nd International Workshop on Computational Terminology (COLING-Workshop)*, pages 29–35, 2002.

[4] Geoff Peters. Geoff's googleduel!, 2002. Available at `http://www.sfu.ca/ gpeters/cgi-bin/pear/`.

[5] Webcorp. Webcorp home page, 1999.

[6] 神部孝. *TOEFL* 英熟語 *850*. 株式会社旺文社, 2001.

[7] 田中久美子, 山本真人, and 中川裕志. web検索に基づく多言語動的KWIC. In **情報処理学会処理研究会**, volume 152, pages 115–122, 2002.