ローカルコーパスからのテキストマイニングツール:PortableKiwi

藤本 宏涼※1　　吉田 稔※2　　清田 陽司※2　　中川 裕志※2

※1 東京大学 大学院情報理工学系研究科　※2 東京大学 情報基盤センター

E-mail: {koufuji,mino,kiyota,nakagawa}@r.dl.itc.u-tokyo.ac.jp

1.はじめに

本論文では、計算機内に蓄積されたローカルなコーパス(以降、ローカルコーパスと呼ぶ)から、文字列単位での多言語用例検索技術Kiwi[1,2]を用いて、コーパスに特有な情報・知識を取り出すテキストマイニングツールPortableKiwiについて述べる。KiwiはWeb上の検索エンジンを用いて表現の用例を調べるツールとして、2003年7月より一般公開されている。※用例を調べたい表現(以降、クエリーと呼ぶ)を入力するとその表現を含むテキストを検索エンジンから動的に得て、そのデータに統計処理を施し用例を表示する(図1)。

しかし現行のKiwiではWeb上の検索エンジンからデータを取得しているため、Web では公開できないような非公開データに対応することができない。また特定の分野に関するデータのみを使用して用例を検索するということも不可能である。さらにKiwiが検索エンジンから得るデータは、検索結果のページに表示されるテキストの一部分のみであるので、テキスト全体を利用しての検索ができない。そこでPortableKiwi ではコーパスをローカルなものとすることで、テキスト全体を用例検索のために利用できるようにした。これにより、分野に特化したコーパスを用いることでその分野に特有な情報を取り出すことができる。またテキストマイニングのための機能強化も行った。

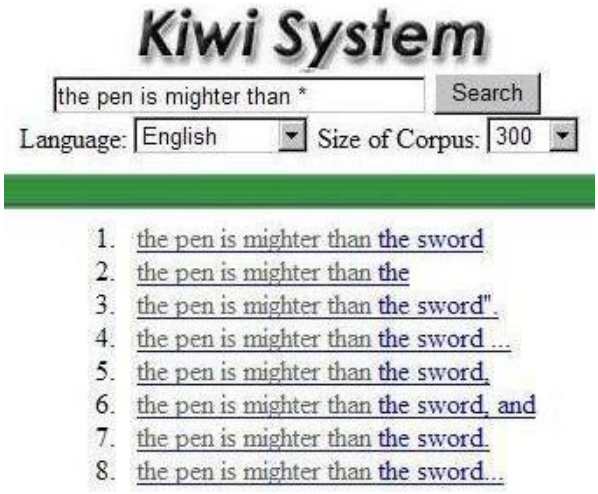

図 1:Kiwi の実行画面

※ http://kiwi.r.dl.itc.u-tokyo.ac.jp/

本論文の構成は以下の通りである。まず、第2節ではKiwiの簡単な説明とローカルコーパスへの対応について示す。第3節ではテキストマイニングのために新たに加えた機能について述べ、第4節で実例を示す。最後に第5節でまとめを述べる。

2.Kiwi のローカルコーパスへの対応

この節ではまず Kiwi のアルゴリズムについて簡単に説明する。その後ローカルコーパスに Kiwi のアルゴリズムを適用するための方法とローカルコーパスを用いるメリットを述べる。

2.1.Kiwi の統計処理

Kiwi は与えたクエリーの前後に来る文字列を以下に述べる方法でスコア付けをし、そのスコア順に表示する。

(1)クエリーを受け取り検索エンジンに送信する。(2)検索エンジンから結果を受け取り、結果中に現れたクエリーの周辺文字列を抜き出し Trie 構造化する。(3)Trie 構造中の文字列から用例の候補を切り出し、スコア付けし順位を与え表示する。クエリーにワイルドカード“*”が含まれている場合、その位置に現れる語を表示する。例えば、“none of your *”というクエリーを記述すると、“business”といった語が表示される。

Kiwi の特徴は候補の切り出しとスコア付けにある。これには“定型的表現中の後続文字種数は後方に進むにつれ減少する”という性質を利用する(図2:矢印は分岐数を示す)。この性

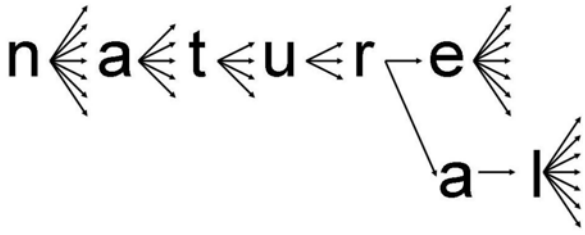

図 2:単語中の分岐数の減少

質を考えると、Trie 構造中において 1 つの定型的表現内では分岐数は減少することとなる。逆に分岐数が増加する箇所は定型的表現の区切りである可能性が高い。

Kiwiではこの性質に基づき結束性の高い定型的表現である文字列Xを以下の式(1)を用いて検出する。ただし、Xは文字列、X_iはその先頭i文字、C_iはX_iに後続する文字種類数すなわちTrieの分岐数である。

$$C_i > C_{i-1} \quad (1)$$

取り出された候補の順位付けには、頻度と長さを用いる。頻度が高く適切な長さを持つものを優先させるために評価関数として以下の式を用いる。(|X|は X の長さを示す。)

$$F(X) = f(X)\log(|X|+1) \quad (2)$$

2.2.ローカルコーパスへの対応

Kiwiにおいては文字列検索のためにTrieを利用しているので、クエリー周辺の文字列データが必要になる。現行の Kiwi では Web 上の検索エンジンを用いて、Web からデータを取り出しているのに対し、PortableKiwi ではローカルに蓄積しているテキストファイルを文字バイグラムでインデクシングし、文字列検索を行う。検索時には与えられたクエリーを文字バイグラムに分割し、それぞれのインデックスを調べることでクエリー周辺文字列データを検索することができる。また 1 文字からなるクエリーの場合は文字バイグラムの 1 文字目が一致するもののインデックスを全て取り出すことで検索することができる。

2.3.メリット

通常のテキストマイニングは単語を単位としての統計処理であるが、Kiwi は文字列単位での用例検索に基づくテキストマイニングであるので、言語に依存しない処理が可能である。またインデックスをローカルコーパスに対して作成することにより、ローカルコーパスの分野に特有な情報を取り出すことができる。例えば、専門分野毎に言葉の使われ方などは異なるので、専門分野ごとのローカルコーパスを使うことで、その分野に特有な言い回しや情報を取り出すことができる。

3.ユーザーインターフェイス

図 3 は PortableKiwi の実行画面である。本ツールでは入力されたクエリーに対する検索結果を順位付けて表示する。表示された結果をクリックすることでそれぞれの結果に対応する原文テキストを閲覧することが可能である。まず Kiwi が提供している機能を述べ、その後に PortableKiwi で新たに加えた機能を述べる。

図 3:PortableKiwi の実行画面

3.1.Kiwi の機能

現行の Kiwi は以下のパターンマッチ機能を提供する。

3.1.1.ワイルドカード　クエリーに“ * ”が含まれている場合その位置に現れる語を表示する。ワイルドカードはクエリーの先頭、末尾、および中間に記述することが可能である。1 つのクエリーに含むことができるワイルドカードの数は 1 つとなっている。

3.1.2.比較検索　用例候補があらかじめいくつかに絞りこまれている場合“ / ”を用いた比較検索が可能である。Kiwi は検索エンジン上でより多く現れる候補から順に並び替えて表示する。

3.1.3.絞込み検索　絞込み語をクエリーに含めることができる。クエリーの末尾に“ +絞込み語 ”と入力することで絞込み語が現れるページ内から情報を検索する。

3.2.PortableKiwi に加えた新機能

PortableKiwi では現行の Kiwi の機能に加え以下の新機能を提供する。

3.2.1.複数ワイルドカード　“ *A* ”いうクエリーを用いると A という関係でつながる前後のテキストを列挙できる。システム内部では、はじめに“ *A ”を検索し、それぞれの結果の末尾に“ * ”をつけ加えもう一度検索する。逆順(はじめに“A* ”)で検索したい場合はクエリーを“ *A*post ”とすればよい。例えば、クエリーとして“ *による* ”を指定する

と原因に対する理由を列挙することができ、また“ *による *post ”とにより理由に対する原因を調べることができる。

3.2.2.絞込み要素の指定　PortableKiwi では絞込みの要素をインデックスに含めておき、ユーザーが検索時に任意の要素を指定できる。例えば、新聞記事をローカルコーパスとして利用する場合インデックスに日付の情報を付け加える。ユーザーはクエリーを“ *事件+2005”とすることで 2005 年のデータのみから情報をとりだすことができる。

3.2.3.pre、post の展開　PortableKiwi では図 3 の実行画面で表示された結果の右の“pre”、“post”をクリックすることでその結果に対し、さらに検索することができる。例えば“ * 問題”の結果の 1 行目で“pre”をクリックすると、“ * BSE 問題”というクエリーを Kiwi に与えた場合の検索結果を新しい window で表示する。“post”をクリックした場合は“BSE 問題*”をクエリーとする検索を行うことになる。

4.航空安全情報における応用例

PortableKiwiによる簡単なテキストマイニングの例として、(株)日本航空インターナショナルにおいて収集されている航空安全レポート※をローカルコーパスとした場合の応用例について述べる。

4.1.前方、中間、後方にワイルドカードを用いる場合

図 4:“による*”の結果

※個人のプライバシー保護のため、個人の特定に繋がる情報は省いてある。航空安全情報は、航空の安全に資するために使用したものである。

図 5:“ *による出発遅延”の結果

まずワイルドカードを 1 つ使って検索する例を示す。実際にどのようなことが発生しているのかを調べるために、“による * ”をクエリーとして検索した結果を図 4 に示す。またさらにその結果から、出発遅延の原因を調べるために“*による出発遅延”をクエリーとして検索した結果を図 5 に示す。これより、作業や旅客数の不一致などにより出発遅延が発生していることがわかる。

4.2.絞込み検索を用いる場合

次に絞込み検索の例を示す。航空安全レポートの場合、絞込みの要素として日付、出発地、到着地をインデックスに含めた。図 6 に羽田空港での出発遅延の原因を調べるためクエリーを“ *による出発遅延+HND ”とした例を示す。ここで“HND”とは羽田空港のことを指す。

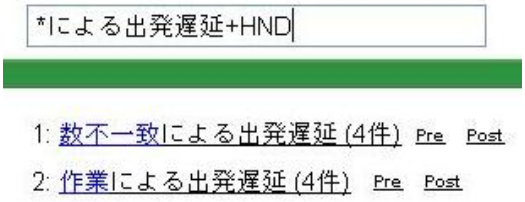

図 6:“ *による出発遅延+HND ”の結果

4.3.複数ワイルドカードを用いる場合

航空安全レポートコーパスを“ *による* ”というクエリーで検索した結果を図 7 に示す。これにより、トラブルの発生パターンを抽出することができる。事前にトラブルの原因を発見することで、対策を立て安全性を向上することができる。さらに絞込み要素を、日付、出発到着の場所などにすることで、より

詳しい原因を調べることができる。詳細な分析については、文献[3]に記載されている。

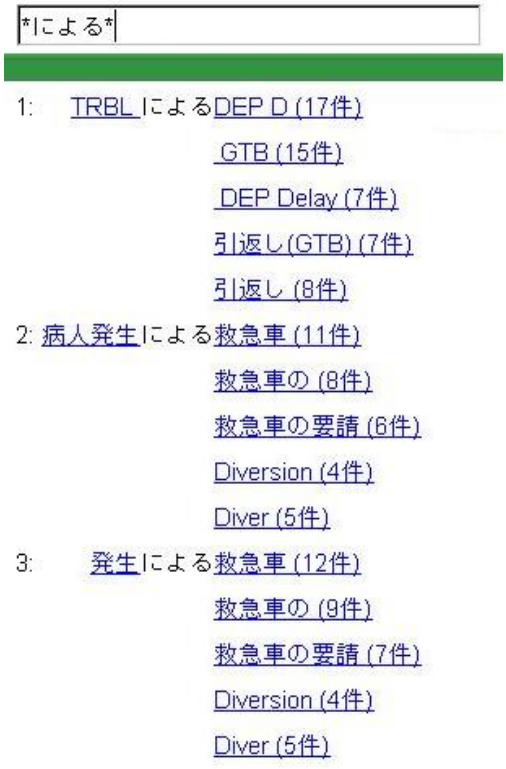

図 7:航空安全レポートの実例 1

図 8:航空安全レポートの実例 2

5.おわりに

本論文では、文字列単位での多言語用例検索技術 Kiwi を基礎に開発した、ローカルコーパス特有な情報･知識を取り出すテキストマイニングツール PortableKiwi を開発し、その実用例を報告した。ローカルコーパスに対し文字バイグラムでインデクシングした検索エンジンを作成することで、コーパス全体から情報を抽出することができ、またその結果に Kiwi の統計処理を行うことで有益な情報を自動的に取り出すことができる。さらにテキストマイニングのための新機能も加えた。今後の課題は①大規模テキストコーパスへの適用、②テキストマイニングのための機能強化、である。

参考文献

[1] Kumiko Tanaka-Ishii, Masato Yamamoto, and Hiroshi Nakagawa. “Kiwi: A multilingual usage consultation tool based on Internet searching” Proceedings of the Interactive Posters/Demonstrations, ACL-03, pages 105-108, 2003

[2] 山本真人, 田中久美子, 中川裕志. “検索エンジンに基づく多言語用例指南ツール:Kiwi ”, 言語処理学会第 9 回年次大会発表論文集, pages 654-657, 2003

[3] 斉藤孝広, 渡部勇, 松井くにお, 寺田昭, 斉藤隆. “航空安全情報からのトラブル発生パターンの抽出について”, 言語処理学会第 11 回年次大会発表論文集, 2005