文化財多言語化のためのPython入門：韓国語の形態素解析

Peter Yanase†
†奈良文化財研究所

An Introductory Course on Python for Translating Cultural Heritage Information: Korean Morphological Analysis

Peter Yanase†
† Nara National Research Institute for Cultural Properties

はじめに

文化財情報のデジタル化とインターネット経由での公開が日々進んでいる中、公開情報をいかに多言語化するかという大きなハードルがデータ管理者を悩ませている。海外向けのサイトを作ったのはいいものの、肝心な検索システムが日本語のままだったりするのはよく見かける光景である。そのため、情報学の専門家たちがここ数十年、専門用語（キーワード）を自動で抽出した多言語辞書データの開発に勤しんでいるが、残念ながら文化財の分野では進展がなかなか見られない。

その一方、近年はPython、R、Rubyなど、比較的に容易に操れるプログラミング言語が広く使われるようになり、数ラインのコードを書くだけで誰もが簡単に自然言語処理が行えるようなってきた。また、同時に、データのオープン化も各国で進み、あらゆる言語で、法的な縛りが大幅に緩和されたテキストデータが入手できるようになった。今はもう自分がぶつかった情報学的な問題を誰かが代わりに解決してくれるのを待つしかない時代ではなくなった。

さて、自然言語処理の多くの場合は分析対象となるコーパスをまず文章、文、単語の順番に区切り、その次または同時に品詞付けを行う。そのため、多くの有志者・研究者がこのような処理ができるツールや辞書データを開発してきた。たとえば、日本語の場合は、MeCab、Janome、ChaSen、JUMAN、KyTeaなどの形態素解析ツールとIPAdic、NAIST-jdic、UniDicなどの形態素解析辞書が存在する。様々なソリューションが開発されているのはもちろん喜ばしいものの、使い分けには一定のリテラシーが必要である。

日本語の解析ツールと辞書であれば、日本人なら特別な知識がなくても、それぞれの解析結果を見比べ、どれが自分の用途に最も合っているのか、比較的に簡単に判断できるであろう。しかし、これは外国語になるとハードルが一気に上がる。なぜなら、なじみがない外国語専用ツールを外国語で書かれているドキュメンテーションを見ながら動かし、外国語で出された解析結果を比較しないといけないからである。

そこで、今回はプログラミング初心者に優しいと評判となっている Python 言語で使える韓国語の自然言語処理ができるライブラリーモジュールを三つ、専門用語・キーワードの抽出に焦点を絞り、簡単に紹介したいと思う。

今回紹介するのは、文境界判定器モジュールの Kss の Python 版、自然言語処理ツール Kiwi の Python ラッパーである kiwipiepy と、自然言語処理ツールキットの KoNLPy の三つである。

環境構築の代わりに

職場や学校のパソコンにセキュリティー上様々な制限がかけられ、自由にソフトウェアをインストールしたり、設定を変えたりすることはできないのが一般的である。そういう場合、Google 社が提供する Web ブラウザー上で使える Python プログラミング環境である「Google Colaboratory」(以降 Colab) が便利である。Colab は Google のアカウントさえあれば、誰でも無料ですぐ使える。パソコンに何もダウンロードやインストールせず、ブラウザー上コードを書き、実行できる。しかも処理スピードもかなり早い。また、Colab が（あえてその環境を構築しない限り）パソコンのファイルに直接アクセスできないのも心強い。ハードディスクのファイルを使ってプログラミングを実行したい場合、まずファイルをアップロードしないといけない。何より、パソコンにインストールされている OS などに依存しないため、Colab を使えば、本記事で紹介する手順はどの環境でも同じになる。

1 使い方

1.1 インストール

Colab なら、３つのモジュールとも pip コマンドを実行するだけで簡単にインストールできる。

Kssの場合は以下のとおりである。

```
pip install kss
```

kiwipiepyの場合は次のようである。

```
pip install kiwipiepy
```

KoNLPyの場合は以下になる。

```
pip install konlpy
```


1.2 インポート

Kssの場合はこれを入力する。

```
import kss
```

kiwipiepyの場合は以下となる。

```
from kiwipiepy import Kiwi
```

KoNLPyにはHannanum、Kkma、Komoran、MecabとOktの５つの形態素解析ツールが含まれ、使用したいものをクラスとしてインポートする必要がある。全部をインポートするなら以下のようになる。

```
from konlpy.tag import Kkma, Hannanum, Komoran, Okt, Mecab
```

Mecab（韓国語版）自体はKoNLPYに含まれていないので、Mecabを使うなら、それが含まれているKssをKoNLPYと一緒にインストールするといい。他のやり方もあるが、これが一番楽である。（Kssをインポートする必要はない。）

1.3 文境界判定

以下は処理対象の文字列の変数を「string」、処理結果の変数を「sentences」と称す。

Kssでのやり方は次のとおりである。

```
sentences = kss.split_sentences(string, backend="mecab")
```

mecabとは別にpecabというバックエンドも用意されているが、mecabの方

が圧倒的に早い。

kiwipiepyでは以下のように入力する。

```
sentences = [sentence.text for sentence in \
               Kiwi().split_into_sents(string)]
```

KoNLPYに含まれているツールの中、Kkmaにしか文境界判定ができない。

```
sentences = Kkma().sentences(string)
```


1.4 単語のトークン化と品詞判定

以下は処理結果の変数を「tokens」と称す。

kiwipiepyの場合は次のように行う。

```
tokens = [(token.form, token.tag) for token in \
            Kiwi().tokenize(string)]
```

KoNLPYに含まれているそれぞれのツールの場合は、以下のようになる。

```
tokens = Kkma().pos(string)
```

```
tokens = Hannanum().pos(string, ntags=22)
```

```
tokens = Komoran().pos(string)
```

```
tokens = Okt().pos(string)
```

```
tokens = Mecab().pos(string)
```

1.5 名詞の抽出

kiwipiepyで固有名詞、普通名詞、代名詞などをまとめて抽出するにはまずそのリストを作成しなければならない。（各種品詞の略語はドキュメンテーションの最後に書いてある。）

```
noun_types = ["NNG", "NNP", "NNB", "NR", "NP"]
```

各種の名詞のリストを作成したら、それをフィルターとして使える。

```
nouns = [token.form for token in Kiwi().tokenize(string) if \
    token.tag in noun_types]
```

KoNLPYに含まれているそれぞれのツールの場合は次のようにする。

```
nouns = Hannanum().nouns(string)
```

```
nouns = Kkma().nouns(string)
```

```
nouns = Komoran().nouns(string)
```

```
nouns = Okt().nouns(string)
```

```
nouns = Mecab().nouns(string)
```


2 比較

2.1 文境界判定

文境界判定のテストには以下のテキストを使った。

> 헤이조쿄는 남북으로 긴 장방형 모양으로 중앙의 주작대로(朱雀大路)를 축으로 우쿄(右京)와 사쿄(左京)로 나뉘어 사쿄의 경사진 면에 게쿄(外京)가 설치되어 있었다. 동서축에는 이치조(一条)에서 구조(九条)까지의 이름이 붙은 대로(시조十条에 대해서는 후술), 남북축에는 주작대로와 사쿄 1 방(坊)에서 4 방, 우쿄 1 방에서 4 방의 큰길이 설치되었던 조방제(条坊制) 형태의 도시계획을 갖추고 있었다. 각 길의 너비는 약 532 미터이고, 길로 에워싸인 부분(방)은 도랑과 쓰이지(築地)에 의해 구획되었고, 나아가 그 안에 동서남북으로 3 개의 길로 구획이 나뉘어 있었다. 헤이조쿄 전체 규모는 동서 길이 약 4.3km(게쿄를 포함하면 6.3km), 남북 약 4.7km(북변의 방은 제외)에 달했다.

上記のテキストをKss、kiwipieとKoNLPYの三つとも正しく4つの文（センテンス）に分けた。ただし、KoNLPYは文の境目を正しく判断したものの、テキストには本来スペースがなかったところに半角スペースを入れた。韓国語においてスペースは文章の意味を変えるほど重要な役割を持っているため、KoNLPYを使った文境界判定はやめた方が無難である。

また、今回はBBSやブログの投稿のような句読点の使い方が独特なテキストの

処理は試していない。そのようなテキストの処理が必要な場合は改めてそれぞれのモジュールの性能を比較するとよい。(ドキュメンテーションを見る限り、そのような文章はKssの方が得意なはずである。)

2.2 名詞抽出

名詞抽出のテストには以下のテキストを使用した。

> 헤이조쿄는 남북으로 긴 장방형 모양으로 중앙의 주작대로(朱雀大路)를 축으로 우쿄(右京)와 사쿄(左京)로 나뉘어 사쿄의 경사진 면에 게쿄(外京)가 설치되어 있었다.

上記の一文に含まれているハングルで書かれている名詞は以下の13語である。

> 헤이조쿄, 남북, 장방형, 모양, 중앙, 주작, 대로, 축, 우쿄, 사쿄, 면, 게쿄, 설치

テスト用のテキストをそれぞれのツールで処理し、上記の名詞のリストと比較したら以下のような結果が得られた。

```
kiwipiepy:
FP(偽陽性): []
FN(偽陰性): ['대로']

Kkma:
FP(偽陽性): ['우', '쿄', '로', '게']
FN(偽陰性): ['대로']

Hannanum:
FP(偽陽性): ['주작대로(朱雀大路)', '우쿄(右京)', '사쿄(左京)', '게쿄(外京)']
FN(偽陰性): ['주작', '대로', '우쿄', '게쿄']

Komoran:
FP(偽陽性): []
FN(偽陰性): ['우쿄', '사쿄', '게쿄']

Okt:
FP(偽陽性): ['를', '로']
FN(偽陰性): ['대로']

Mecab:
FP(偽陽性): ['朱雀', '大路', '右', '京', '左', '京', '게', '쿄', '外', '京']
FN(偽陰性): ['대로', '게쿄']
```

まとめ

今回は文化財多言語化の過程における韓国語の自然言語処理に役立つ Python モジュールをを三つを簡単に紹介した。むろん、それぞれには今回言及しなかった機能や詳細な設定が存在するが、基本的な使い方は紹介できたかと思う。最後にはそれぞれを使ってみて思ったことを簡略にまとめる。

Kss は文境界判定に特化した分、コマンドが分かりやすく、スピードが速く、正確である。

kiwipiepy は文境界判定に加え、単語のトークン化と品詞付けもでき、名詞の抽出の実験では最も正確な判断をした。しかし、Python 初心者には少し使いにくい。

KoNLPY の文境界判定は使わない方が無難であるが、形態素解析ツールは複数用意され、コマンドが統一されて分かりやすい。

用途にもよるので、一概にどれがいいとは言えない。また、それぞれの強みを生かすため、組み合わせて使ってもよい。

参考文献

Ko, Hyunwoong, Sang-kil Park. Kss: A Toolkit for Korean sentence segmentation (v. 4.5.1) [Source Code]. *https://github.com/hyunwoongko/kss*

Lee, Minchul. kiwipiepy (v. 0.14.1) [Source Code]. *https://github.com/bab2min/kiwipiepy*

Park, Eunjeong L. and Sungzoon Cho. “KoNLPy: Korean natural language processing in Python,” Proceedings of the 26th Annual Conference on Human & Cognitive Language Technology, Chuncheon, Korea, Oct 2014.

KoNLPy (v. 0.6.0) [Source Code]. *https://github.com/konlpy/konlpy*