Building your KoNLPy

(Python 에서 다른 언어 이용하기)

Hyunjoong Kim

soy.lovit@gmail.com

github.com/lovit

- Python 외의 언어로 구현된 한국어 형태소 분석기들이 많습니다.
  - 알고리즘을 Python 으로 재구현하는 것보다, Jpype 와 같은 라이브러리를 이용하여 Python 환경에서 다른 언어의 라이브러리를 이용할 수 있습니다.
  - KoNLPy 는 이 기능을 제공합니다.

- 다른 언어로 구현된 알고리즘을 수정한 뒤, 이를 Python 에서 이용하기 위해서는 다른 언어와 Python 을 연결하는 라이브러리를 다룰 수 있어야 합니다.

Java in Python
(Komoran 3 in Python)

Komoran 3

- Shin Junsoo [1] 님이 Java 로 구현한 한국어 형태소 분석기 입니다.

```
import java.util.List;
import kr.co.shineware.nlp.komoran.constant.DEFAULT_MODEL;
import kr.co.shineware.nlp.komoran.core.Komoran;
import kr.co.shineware.nlp.komoran.model.Token;

public class CustomizedTest {
    public static void main(String[] args) {
        Komoran komoran = new Komoran(DEFAULT_MODEL.FULL);
        List<Token> tokens = komoran.analyze("청하는아이오아이출신입니다").getTokenList();
        for(Token token : tokens)
            System.out.println(token);
    }
}
```


[1] https://github.com/shin285

Komoran 3

- Shin Junsoo [1] 님이 Java 로 구현한 한국어 형태소 분석기 입니다.
  - 빠른 속도로 형태소 분석을 수행합니다.
  - 미등록단어 문제가 발생할 수 있습니다.

```
List<Token> tokens = komoran.analyze("청하는아이오아이출신입니다").getTokenList();
```

```
Token [morph=청하, pos=VV, beginIndex=0, endIndex=2]
Token [morph=는, pos=ETM, beginIndex=2, endIndex=3]
Token [morph=아이오, pos=NNG, beginIndex=3, endIndex=6]
Token [morph=아이, pos=NNG, beginIndex=6, endIndex=8]
Token [morph=출신, pos=NNG, beginIndex=8, endIndex=10]
Token [morph=이, pos=VCP, beginIndex=10, endIndex=11]
Token [morph=ㅂ니다, pos=EC, beginIndex=10, endIndex=13]
```


[1] https://github.com/shin285

Komoran 3

- Shin Junsoo [1] 님이 Java 로 구현한 한국어 형태소 분석기 입니다.
  - 사용자 사전을 추가할 수 있습니다.

```
komoran.setUserDic("user_data/dic.user");
List<Token> tokens = komoran.analyze("청하는아이오아이출신입니다").getTokenList();
```

```
Token [morph=청하, pos=VV, beginIndex=0, endIndex=2]
Token [morph=는, pos=ETM, beginIndex=2, endIndex=3]
Token [morph=아이오아이, pos=NNG, beginIndex=3, endIndex=8]
Token [morph=출신, pos=NNG, beginIndex=8, endIndex=10]
Token [morph=이, pos=VCP, beginIndex=10, endIndex=11]
Token [morph=ㅂ니다, pos=EC, beginIndex=10, endIndex=13]
```


[1] https://github.com/shin285

Komoran 3

- 사용자 사전은 tap separated 형식의 텍스트 파일입니다.
  - <단어, 품사> 형태로 입력합니다.
  - 단어는 “바람과 함께 사라지다” 처럼 띄어쓰기가 포함될 수 있습니다. 구문을 하나의 명사로 취급할 수 있습니다.

```
# 사용자 사전.
# 은 주석
# tap separated <단어, 품사>

아이오아이 NNP
바람과 함께 사라지다 NNP
```

Komoran 3

- 각 토큰의 정보를 가져올 수 있습니다.

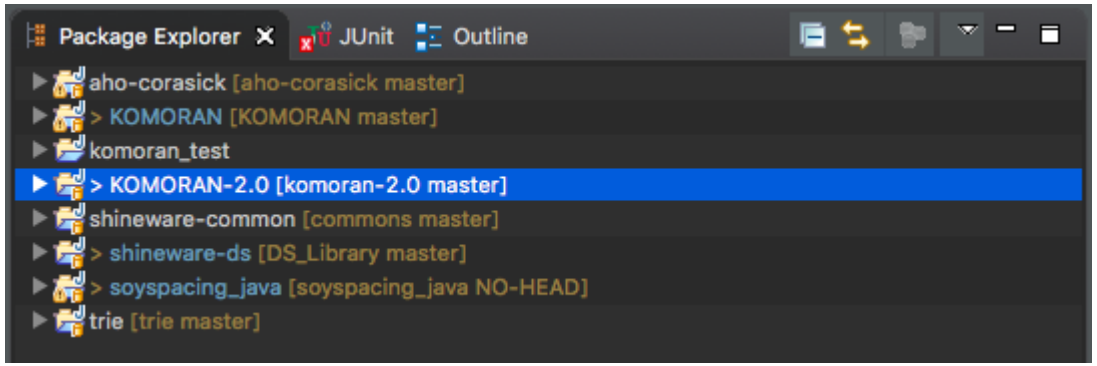
```
Token [morph=청하, pos=VV, beginIndex=0, endIndex=2]
Token [morph=는, pos=ETM, beginIndex=2, endIndex=3]
…
```

```
tokens.get(0).getBeginIndex(); // int
tokens.get(0).getEndIndex(); // int
tokens.get(0).getMorph(); // String
tokens.get(0).getPos(); // String
```

Eclipse 를 이용하여 Java Project 를 JAR 로 만들기

- Java IDE 중 하나인 Eclipse 를 이용하여 JAR 파일을 만들 수 있습니다.
  - JAR 는 클래스 파일과 데이터 파일이 합쳐진 Java ARchive 입니다.

- Eclipse 의 Package Explorer 에서 JAR 로 만들 프로젝트를 우클릭 합니다.
  - Export → Java → JAR file 을 누릅니다.



Jpype

- Jpype 는 Python kernel 에 JVM 을 띄워서 Python 에서 Java 를 이용할 수 있도록 도와주는 라이브러리 입니다.

Jpype 설치하기

- Jpype 는 아래의 명령어로 설치가 가능합니다.

  > pip install Jpype2

- Pypi 에 등록된 이름과 package 이름이 다르니 주의합니다.

Jpype 로 JVM 띄우기

- JVM 을 띄울 때 네 가지 정보를 설정합니다.
  - JVM path : 자바 설치 위치
  - class path : 사용할 라이브러리
  - Default encoding
  - max heap memory size

Jpype 로 JVM 띄우기

- Java 가 설치된 위치는 jpype 에서 찾을 수 있습니다.
- Default Java 가 아닌 다른 Java 를 이용하려면 임의로 설정하면 됩니다.

```
jvmpath = jpype.getDefaultJVMPath()
```

Jpype 로 JVM 띄우기

- JVM 에서 이용할 라이브러리들의 “절대주소” 를 모두 classpath 로 입력해야 합니다. 절대주소는 os.path 를 이용하여 알 수 있습니다

```
import os

installpath = os.path.dirname(os.path.realpath(__file__))
libpaths = [
    '{0}',
    '{0}{1}bin',
    '{0}{1}aho-corasick.jar',
    '{0}{1}shineware-common-1.0.jar',
    '{0}{1}shineware-ds-1.0.jar',
    '{0}{1}komoran-3.0.jar',
    '{0}{1}*'
]
javadir = '%s%sjava' % (installpath, os.sep)
args = [javadir, os.sep]
libpaths = [p.format(*args) for p in libpaths]
classpath = os.pathsep.join(libpaths)
```

Jpype. One JVM in one Python kernel

- Jpype 는 하나의 Python kernel 에 하나의 JVM 을 띄웁니다.
  - 두 개 이상의 JVM 을 띄우려 하면 오류가 납니다.
  - JVM 이 띄워져 있는지 확인한 뒤, 안전하게 JVM 을 실행합니다.

```
import jpype


if jpype.isJVMStarted():
    try: jpype.startJVM(
        jvmpath,
        '-Djava.class.path=%s' % classpath,
        '-Dfile.encoding=UTF8',
        '-ea',
        '-Xmx{}m'.format(max_memory)
        )
    except Exception as e:
        print(e)
```

Jpype 로 JVM 띄우기

- 앞의 내용을 정리합니다.
- JVM 을 띄우는 Python 함수, init_jvm() 을 만듭니다.

```
import os, sys, jpype
Def init_jvm(libraries=None, max_heap=1024):
    if jpype.isJVMStarted():
        return None
    if not libraries:
        installpath = os.path.dirname(os.path.realpath(__file__))
        libpaths = ['{0}', '{0}{1}bin', '{0}{1}aho-corasick.jar',
                    '{0}{1}shineware-common-1.0.jar',
                    '{0}{1}shineware-ds-1.0.jar',
                    '{0}{1}komoran-3.0.jar', '{0}{1}*' ]
        javadir = '%s%sjava' % (installpath, os.sep)

    args = [javadir, os.sep]
    libpaths = [p.format(*args) for p in libpaths]
    classpath = os.pathsep.join(libpaths)
    jvmpath = jpype.getDefaultJVMPath()
    try:
        jpype.startJVM(
            jvmpath,
           '-Djava.class.path=%s' % classpath,
           '-Dfile.encoding=UTF8',
           '-ea', '-Xmx{}m'.format(max_heap)
        )
    except Exception as e:
        print(e)
```

Komoran Python class 만들기

- init_jvm() 을 이용하여 JVM 을 띄운 뒤에는 Java 처럼 이용하면 됩니다.

```
import os, jpype
from .jvm import init_jvm

class Komoran:
    def __init__(self):
        init_jvm()
        package = jpype.JPackage('kr.co.shineware.nlp.komoran.core')
        model_path = os.path.dirname(os.path.realpath(__file__)) + '/models/'
        self._komoran = package.Komoran(model_path)

    def set_user_dictionary(self, path):
        self._komoran.setUserDic(path)

    def pos(self, sent):
        tokens = self._komoran.analyze(sent).getTokenList()
        tokens = [(token.getMorph(), token.getPos()) for token in tokens]
        return tokens
```

Customize Komoran

- Komoran 에 입력되는 input 에 전처리를 하거나, 형태소 분석된 결과에 후처리를 한 Java code 를 만듭니다. 혹은 Komoran 의 일부를 각자의 상황에 맞게 튜닝합니다.
- 위의 방법처럼 이를 JAR 파일로 만든 뒤, Jpype 로 Python 과 연결하면 Komoran 을 Java 에서 customized 할 수 있습니다.
- 물론, Komoran 의 core 를 변경하는 것이 아니라면, Python 에서 전/후처리를 할 수도 있습니다.