技術報告

afterKNP

KNP の出力を JSON に変換するシステム

中村勝則

2018 年 12 月 01 日

目 次

1 afterKNP

日本語文章の係り受け構造を解析するツールである KNP の出力を受けて，意味の抽出に適した JSON オブジェクトを生成するシステム **afterKNP** について説明する．afterKNP は Python 言語で記述されており，Python 処理系のためのモジュールとして構築されている．

1.1 JUMAN と KNP

KNP は日本語文章の係り受け構造を解析するツールである．KNP は **JUMAN**（日本語文章の形態素を解析するツール）の解析結果を入力とする．これらは京都大学で開発されたツール [1] であり，OS のコマンドライン（コマンドシェル）で使用する．

【JUMAN,KNP の使用例】

例えば次のようなテキストファイルの内容を解析することを考える．

日本語文書のテキストファイル：knptest01.txt

```
1 大阪の太郎と東京の花子が名古屋のレストランで食事をした。
```

このテキストを JUMAN → KNP を経て解析する操作は次のように行う．

（Windows 環境での例）`juman < knptest01.txt | knp`

この結果，次のような内容が標準出力に出力される．

KNP の出力：

```
1 大阪の──┐
2         太郎と(P)─┐
3 東京の──┐         │
4         花子が(P)─PARA──┐
5       名古屋の──┐       │
6           レストランで──┤
7                 食事を──┤
8                         した。
```

このように文章の係り受けが木構造として得られる．KNP 実行時に ‘-tab’ オプションを与えると，木構造の表現ではなく 形態素情報のデータレコード が得られ，他のプログラムによる文章構造解析に利用することができる．

KNP が出力する形態素情報は，係り受けの深さの数値や各品詞に関する豊富な情報を含んでおり，日本語文章が持つ 意味に関する情報 を取り出すのに役立つ．

1.2 MSO（Modification Structure Object）

afterKNP は KNP の出力から意味に関する情報を **MSO**（Modification Structure Object）として取得する．MSO は JSON 形式のオブジェクトであり，先のテキストファイル knptest01.txt の意味情報を持つ MSO を生成する．（次の例）

[1] 京都大学 大学院情報学研究科 黒橋・河原研究室（http://nlp.ist.i.kyoto-u.ac.jp/）

MSO：“大阪の太郎と東京の花子が名古屋のレストランで食事をした。”

```
[{'apdx': [{'text': 'が',
            'wc1': '助詞',
            'wc2': '格助詞'}],
  'feature': {'cp': 'が',
              'type': 'subject'},
  'mod': [{'apdx': [{'text': 'の',
                     'wc1': '助詞',
                     'wc2': '接続助詞'}],
           'feature': {'cp': 'の',
                       'type': 'adj'},
           'text': '東京',
           'wc1': '名詞',
           'wc2': '地名'},
          {'apdx': [{'text': 'と',
                     'wc1': '助詞',
                     'wc2': '格助詞'}],
           'feature': {'cp': 'と',
                       'type': ['cp',
                                'と']},
           'mod': [{'apdx': [{'text': 'の',
                              'wc1': '助詞',
                              'wc2': '接続助詞'}],
                    'feature': {'cp': 'の',
                                'type': 'adj'},
                    'text': '大阪',
                    'wc1': '名詞',
                    'wc2': '地名'}],
           'text': '太郎',
           'wc1': '名詞',
           'wc2': '人名'}],
  'text': '花子',
  'wc1': '名詞',
  'wc2': '人名'},
 {'apdx': [{'text': 'で',
            'wc1': '助詞',
            'wc2': '格助詞'}],
  'feature': {'cp': 'で',
              'type': ['cp',
                       'で']},
  'mod': [{'apdx': [{'text': 'の',
                     'wc1': '助詞',
                     'wc2': '接続助詞'}],
           'feature': {'cp': 'の',
                       'type': 'adj'},
           'text': '名古屋',
           'wc1': '名詞',
           'wc2': '地名'}],
  'text': 'レストラン',
  'wc1': '名詞',
  'wc2': '普通名詞'},
 {'apdx': [{'text': 'を',
            'wc1': '助詞',
            'wc2': '格助詞'}],
  'feature': {'cp': 'を',
              'type': 'object'},
  'text': '食事',
  'wc1': '名詞',
  'wc2': 'サ変名詞'},
 {'apdx': [{'text': '。',
            'wc1': '特殊',
            'wc2': '句点'}],
  'feature': {'affm': True,
              'type': 'verb'},
  'text': 'した',
  'wc1': '動詞',
  'wc2': '*'}]
```

1.2.1 MSO の基本構造

一階述語論理では，**対象**が行う**動作**や，**対象**の**属性値**を，**論理式**（節）として表現する．afterKNP は日本語文章を一階述語論理の式（特に Prolog 言語のホーン節）に変換するための前処理を想定して MSO を生成する．そのため MSO は，**名詞**, **動詞**, **形容詞**, **副詞**から成るリストとして構成される．例えば先の日本語文章

“大阪の太郎と東京の花子が名古屋のレストランで食事をした。”

は MSO の要素として**花子**, **名古屋**, **した**の 3 つに要約（リストとして要約）され，元の文章を

“**花子**が**名古屋**で食事を**した**．”

と表現する．またその他の要素は MSO の各要素の**係り受け部**として保持される．例えば「花子」の係り受け部として「東京の」が保持される．

【MSO の構造の詳細】

- **基本構造**
  MSO は**リスト**（列）である．MSO の要素は**辞書**（連想オブジェクト）である．

- **MSO の要素**
  MSO の要素は辞書であり，次のようなキーに各種情報が保持される．
  - **‘text’** ： MSO 要素の**幹**（品詞を決定する部分）となる語の文字列を保持する．
  - **‘wc1’,‘wc2’** ： 品詞情報．品詞名の文字列を保持する．
  - **‘apdx’** ： MSO 要素の幹以外の部分（接尾辞や助詞など）を保持する**リスト**．この要素は MSO 要素に倣った辞書オブジェクトであり，‘text’, ‘wc1’,‘wc2’ のキーを持つ．
  - **‘mod’** ： 係り受けを保持する**リスト**．複数の係り受け部をリストの要素として持つ．このリストの要素は MSO 要素に倣った辞書オブジェクトであり，‘text’,‘wc1’,‘wc2’, ‘apdx’,‘mod’ といったキーを持つ．（再帰的な係り受けを可能とする）
  - **‘feature’** ： 特徴データ．辞書オブジェクトであり，‘type’,‘cp’ のキーを持つ．（詳細は後述）

- **MSO の特徴データ**
  MSO の要素は ‘feature’ をキーとする特徴データを持つ．特徴データは，当該 MSO 要素が持つ役割を知る手がかりを与えるものである．役割とは例えば，その要素が主語になるのか，あるいは述語になるのかとったことを意味する．特徴データのキーとそれに対する値としては次のようなものがある．
  - **‘type’** ： `‘subject’/‘object’/‘pred’/‘adj’/‘verb’/‘cond’/[‘cp’,` 要素リスト`]`
    ‘subject’ →主語になる可能性がある， ‘object’ →目的語になる可能性がある．
    ‘pred’ →述語（〜である）になる可能性がある． ‘adj’ →他の要素を修飾する可能性がある．
    ‘verb’ →動詞． ‘cond’ →条件（〜なら）． [‘cp’, 要素リスト] →付加的要素（〜で,〜と, 他）
  - **‘affm’** ： `True/False` →要素が肯定的／否定的かについての示唆
  - **‘tense’** ： `-1/0/1` （時制に付いての示唆）
    -1 →過去， 0 →現在， 1 →未来
  - **‘cp’** ： 助詞（格助詞：〜は,〜が）

  上記の内，‘affm’,‘tense’,‘cp’ が得られない場合もある．

1.3 処理の概要

afterKNP は主に次の 2 つの関数を提供する．

- **JUMAN,KNP を起動して日本語文書を処理する関数：**jparseKnp
  この関数は OS のコマンドとして JUMAN，KNP を起動して日本語文章を処理する．結果としての戻り値は ‘knp -tab’ のコマンド処理の結果を保持するリストである．このリストは KNP の出力の各レコードを要素とする．この際，各レコードを項目毎に分割し，それらを要素として持つリストとしている．
- **KNP の出力から MSO を生成する関数：**janalyze

1.3.1 janalyze の動作

janalyze はまず最初に，KNP の出力に含まれている情報を元にして，文章を構成する各語の係り受けの階層関係を再帰的なリスト（**係り受け構造リスト**：MSL [2] ）として構築する．次に，得られた MSL から品詞の情報を元にして MSO を生成する．

2 使用方法

afterKNP は Python 処理系で使用するモジュールとして構築されているが，これの使用には，JUMAN と KNP, それに **nkf** [3] が OS のコマンドとして使用できることが前提となる．

■ KNP による係り受け解析の実行

関数の呼び出し方： jparseKnp(**日本語文章の文字列**)

この呼び出しにより JUMAN, KNP, NKF が順に起動され，KNP による係り受け解析が実行される．
戻り値は，KNP の出力をリストの形式に変換したものである．

■ MSO の生成

関数の呼び出し方： janalyze(jparseKnp **の戻り値**)

この呼び出しにより MSO が戻り値として生成される．

例. afterKNP の使用

```
>>> import afterKNP as ak  Enter     ← afterKNP モジュールの読込み
>>> s = ’大阪の太郎と東京の花子が名古屋のレストランで食事をした。’  Enter     ←日本語文章の用意
>>> L0 = ak.jparseKnp( s )  Enter     ← KNP の処理を実行
>>> L1 = ak.janalyze( L0 )  Enter     ← MSO の生成
>>> from pprint import pprint  Enter     ← pprint モジュールの読込み
>>> pprint(L1,width=10,indent=1)  Enter     ← MSO の表示
[{’apdx’:  [{’text’:  ’が’,
             ’wc1’:  ’助詞’,
             ’wc2’:  ’格助詞’}],
  ’feature’:  {’cp’:  ’が’,
               ’type’:  ’subject’}
                  :
               （以下省略）
                  :
```

[2] **係り受け構造リスト**：MSL（Modification Structure List） — afterKNP 独自の日本語文書表現.
[3] **nkf**（Network Kanji Filter）：多バイト文字の文字コードを変換するための有名なツール.

索 引