python-asa2prolog

Description

ASAの解析結果をPrologの木構造に変換する

環境構築(WIP)

取り急ぎ必要な環境をまとめておきます。後ほど修正。

- python >=3.6.8
- Mecab && Cabocha
- Graphviz
- numpy
- regex
- 必要物のインストール
  - `$ pip install -r requirements.txt`
  - `$ git clone https://github.com/takenl2021/python_prolog_interpreter.git`

asa2prolog_converter.Converter

以下、コンバータのインターフェース

メソッド	説明	引数	戻り値
set_sentences()	生テキストのセット	引数１[string]: 生のテキスト	void
load_sentences()	ファイルからテキストのロード	引数１[string]: ファイルパス	void
get_sentences()	セットされているテキストリスト取得	None	string[]
convert()	一文をコンバート	引数１[string]: 生のテキスト	{ 'predicates': string(一文に対するProlog述語), 'dot_string': string(DOT), 'asa_json': dict(ASAの出力JSON) }
convert_all()	ロードされている全文をコンバート	None	{ 'predicates': string(一文に対するProlog述語), 'dot_string': string(DOT), 'asa_json': dict(ASAの出力JSON) }[]

対応述語

以下、生成されるProlog述語一覧

述語名	引数１	引数２	引数３
chunk(_ , _ , _)	文番号	0固定	chunkノード番号
morph(_ , _ , _)	文番号	親chunkノード番号	morphノード番号
main(_ , _ , _)	文番号	親chunkノード番号	親chunkの主形態素の表層
part(_ , _ , _)	文番号	親chunkノード番号	親chunkの副形態素の表層
role(_ , _ , _)	文番号	親chunkノード番号	親chunkの意味役割の表層
semantic(_ , _ , _)	文番号	親chunkノード番号	親chunkの概念の表層
surf(_ , _ , _)	文番号	0/chunk/morphノード番号	ノードの表層
surfBF(_ , _ , _)	文番号	morphノード番号	ノードの表層の基本形
sloc(_ , _ , _)	文番号	chunk/morphノード番号	ノードの表層のsloc

ルールの設定

使用したいルールをあらかじめ定義しておくことで、解探索の際にそのルールを使用することができる。

`config/rules.pl`の形式で保持しておく。

ルールのロード/使用は`main.py`に例あり。

探索結果の整形

探索結果はdefaultdict形式で返る。dictにすると以下の形式。

(例) クエリ `something(X,Y,Z)`を実行したとする

```
{
  'X': [一番目の解のX, 二番目の解のX, 三番目の解のX, ...],
  'Y': [一番目の解のY, 二番目の解のY, 三番目の解のY, ...],
  'Z': [一番目の解のZ, 二番目の解のZ, 三番目の解のZ, ...],
}
```

この形式だと処理しにくいため、

```
[
  [一番目の解のX, 二番目の解のX, 三番目の解のX, ...],
  [一番目の解のY, 二番目の解のY, 三番目の解のY, ...],
  [一番目の解のZ, 二番目の解のZ, 三番目の解のZ, ...],
]
```

上記のように二次元配列化後に転置、辞書化する例を`main.py`に記載した。整形後は以下の形式。

```
[
  {
    'X': 一番目の解のX,
    'Y': 一番目の解のY,
    'Z': 一番目の解のZ,
  },
  {
    'X': 二番目の解のX,
    'Y': 二番目の解のY,
    'Z': 二番目の解のZ,
  },
  {
    'X': 三番目の解のX,
    'Y': 三番目の解のY,
    'Z': 三番目の解のZ,
  },
  ...
]
```

もう少しうまく書けそうな気がするので、思い付けば共有お願いします。