プロジェクト第４回

形態素解析

形態素解析

形態素解析

- テキストから文法や辞書と呼ばれる単語の品詞などの情報に基づき、形態素の列に分割し、それぞれの形態素の品詞などを判別する作業

- 形態素：言語で意味をもつ最小単位

MeCabファイル

- “吾輩は猫である”のMeCabファイルを作成する。

```
['一\t名詞,数,*,*,*,*,一,イチ,イチ',
 '\u3000\t記号,空白,*,*,*,*,\u3000,\u3000,\u3000',
 '吾輩は猫である\t名詞,固有名詞,一般,*,*,*,吾輩は猫である,ワガハイハネコデアル,ワガハイワネコデアル',
 '。\t記号,句点,*,*,*,*,。,。,。',
 '名前\t名詞,一般,*,*,*,*,名前,ナマエ,ナマエ',
 'は\t助詞,係助詞,*,*,*,*,は,ハ,ワ',
 'まだ\t副詞,助詞類接続,*,*,*,*,まだ,マダ,マダ',
 '無い\t形容詞,自立,*,*,形容詞・アウオ段,基本形,無い,ナイ,ナイ',
 '。\t記号,句点,*,*,*,*,。,。,。',
 '\u3000\t記号,空白,*,*,*,*,\u3000,\u3000,\u3000',
 'どこ\t名詞,代名詞,一般,*,*,*,どこ,ドコ,ドコ',
 'で\t助詞,格助詞,一般,*,*,*,で,デ,デ',
 '生れ\t動詞,自立,*,*,一段,連用形,生れる,ウマレ,ウマレ',
 'た\t助動詞,*,*,*,特殊・タ,基本形,た,タ,タ',
 'かとん\t名詞,一般,*,*,*,*,火遁,カトン,カトン']
```

MeCabファイルの作成

- neko.txtを以下のリンクにアクセスしてダウンロードする。

http://www.cl.ecei.tohoku.ac.jp/nlp100/data/neko.txt

- 次のページのコードを実行するPythonファイルと同じ位置にneko.txtを置く。

MeCabファイルの作成

```
import MeCab

def make_mecab_file(input_file_name, output_file_name):
    m = MeCab.Tagger()
    with open(input_file_name, encoding='utf-8') as input_file:
        with open(output_file_name, mode='w', encoding='utf-8') as output_file:
            output_file.write(m.parse(input_file.read()))

make_mecab_file("neko.txt", "neko.txt.mecab")
```

MeCabファイルの読み込み(1/2)

```
import re
with open(“./neko.txt.mecab”, “r”, encoding=“utf-8”) as fp:
    col=[]
    sent=[]
    Keys=[“surface”, ”base”, “pos”, “pos1”]
    for line in fp:
        Values=[]
        words=re.split(“¥t|,|¥n”, line)
        if words[0]==“EOS”:
            if sent:
                col.append(sent)
                sent=[]
            continue
         Values.append(words[0])    # 単語を格納
```

MeCabファイルと同じ場所に実行するPythonファイルを置く

surface：表層形
base：基本形
pos：品詞
pos1：品詞細分類

MeCabファイルの読み込み(2/2)

```
        Values.append(words[0])  # 単語を格納
        Values.append(words[7])
        Values.append(words[1])
        Values.append(words[2])
        sent.append(dict(zip(Keys,Values)))

print(col)
```

動詞の抽出

```
verb_sur=[]
for d in col:
    for sen in d:
        if sen[“pos”]==“動詞”:
                verb_sur.append(sen[“surface”])

print(verb_sur)
```

演習1. 動詞の原形抽出

MeCabファイルを読み込み、動詞の原形を抽出せよ。前スライドを参考にせよ。

演習2. サ変名詞の抽出

MeCabファイルを読み込み、サ変名詞を抽出せよ。サ変名詞の場合、pos1は”サ変接続”であり、baseは”*”でない。

演習3. AのB

2つの名詞が「の」で連結されている名詞句を抽出せよ。「の」の前後のposは”名詞”である。

単語の出現頻度

文章中に出現する単語とその出現頻度を求め、出現頻度の高い順に並べる。

```
import collections


word_freq=collections.Counter()
for d in col:
    for sen in d:
        word_freq.update(sen[“surface”])
print(word_freq.most_common())
```

matplotlibのインストール

- 管理者コマンドプロンプトで

  pip install matplotlib

  または

  python -m pip install matplotlib

- インストールができたかの確認は次スライドのコードで行う。エラーが出なければOK。

出現頻度の可視化

単語の出現頻度のヒストグラム（横軸：出現頻度, 縦軸：単語の種類数）を描く。

```
from matplotlib import pyplot as plt
counts=list(zip(*word_freq.most_common()))[1]
plt.xlabel(“word-frequency”)
plt.ylabel(“word-kinds”)
plt.hist(counts,bins=20,range=(1,20),normed=True)
plt.xlim(xmin=1,xmax=20)
plt.grid(axis=“y”)
plt.show()
```

Zipfの法則

単語の出現頻度順位を横軸、その出現頻度を縦軸として、両対数グラフをプロットする。

```
from matplotlib import pyplot as plt
counts=list(zip(*word_freq.most_common()))[1]
plt.scatter(range(1, len(counts)+1),counts)
plt.xlim(1,len(counts)+1)
plt.ylim(1,counts[0])
plt.xscale(“log”)
plt.yscale(“log”)
plt.grid(axis=“both”)
plt.show()
```

演習4. 可視化の分析

前スライドのグラフから、何が読み取れるか。