Семантические сети

Лебедев Андрей, группа 210

Подключения:

```
from nltk.corpus import wordnet as wn
```

Из модуля nltk подключаем корпус WordNet для анализа связей слов в наборе тестов WordSim353, а именно файле wordsim353_agreed.txt.

План работы:

```
f = open('wordsim353.txt', 'r')
```

Подключаем набор тестов, а далее для каждой пары проверяем, являются ли слова в ней синонимами или связаны отношением гипоним-гипероним:

```
for s in f:
   if s[0] != '#':
       type, word1, word2, evaluation = s.split()
       if are_synonyms(word1, word2):
           print(word1, word2, file = f_syn)
           synonyms_ev += float(evaluation)
           synonyms_cnt += 1
       elif are_h_nyms(word1, word2):
           print(word1, word2, file = f_h_onyms)
           h_nyms_ev += float(evaluation)
           h_nyms_cnt += 1
       else:
           print(word1, word2, file = f_other)
           other_ev += float(evaluation)
           other_cnt += 1
```

Здесь word1, word2 – пара слов, evaluation – вес близости. Переменные с постфиксом ev и cnt служат для вычисления среднего арифметического веса близости каждой группы слов.

Описания функций are_synonyms и are_h_nyms:

```
def are_synonyms(word1, word2):
    lst1 = wn.synonyms(word1)
    lst2 = wn.synonyms(word2)
    flag = False
    for i in lst1:
        if word2 in i:
            flag = True
            break
    if flag == True:
        flag = False
        for i in lst2:
            if word1 in i:
                return 1
    return 0
```

Функция synonyms возвращает вложенный список синонимов, сгруппированных по значению. Для проверки, является ли пара слов синонимами убеждаемся, что первое слово входит в множество синонимов второго и наоборот.

```
def are_h_nyms(word1, word2):
    lst1 = wn.synsets(word1)
    lst2 = wn.synsets(word2)
    if len(lst1) != 0 and len(lst2) != 0:
        word1 = lst1[0]
        word2 = lst2[0]
        hypoword1 = set([i for i in word1.closure(lambda s:
s.hyponyms())])
        hypoword2 = set([i for i in word2.closure(lambda s:
s.hyponyms())])
        hyperword1 = set([i for i in word1.closure(lambda s:
s.hypernyms())])
        hyperword2 = set([i for i in word2.closure(lambda s:
s.hypernyms())])
        return word1 in hypoword2 or word1 in hyperword2 or word2 in
hypoword1 or word2 in hyperword1
    else:
        return 0
```

Здесь получаем синсет для каждого из двух слов и проверяем, содержится ли он в множестве гипонимов или гиперонимов другого слова.

Результаты:

1) Пары синонимов:

   wood forest
   king queen
   car automobile
   gem jewel
   magician wizard
   midday noon
   calculation computation
   dollar buck

   Количество пар: **8**
   Средний вес связи: **8.7725**

2) Пары слов со связью гипоним-гипероним:

   media radio
   fuck sex
   football soccer
   coast shore
   tool implement
   tiger organism
   psychology science
   psychology discipline
   psychology cognition
   precedent example
   precedent cognition
   cup tableware
   cup artifact
   cup object
   cup entity
   type kind
   phone equipment
   seafood lobster
   environment ecology
   aluminum metal

   Количество пар: **20**
   Средний вес связи: **6.5849**

3) Другие типы отношений:

   love sex
   tiger cat

tiger tiger
book paper
computer keyboard
computer internet
plane car
train car
telephone communication
television radio
drug abuse
bread butter

И так далее, полный список см. в файле other.txt.

Количество пар: **324**
Средний вес связи: **5.7288**

Выводы:

В ходе работы:

1) Познакомились с пакетом nltk доступа к WordNet;
2) Определили механизм взаимодействия с синсетами;
3) Узнали о том, как проверить, являются ли слова синонимами или связаны ли они как гипоним-гипероним.

Изучив документацию к пакету nltk, узнали о функции lemmas и о ее отличиях от synsets, а также о различных способах вычисления сходства слов, таких как Path Similarity, Leacock-Chodorow Similarity, Wu-Palmer Similarity, Resnik Similarity Jiang-Conrath Similarity и Lin Similarity.