Practical exercise 4 - experiments with Wordnet

WordNet is a large digital lexicon made by hand. The kernel of WordNet are the so called synsets that can be understood as meanings. Each word belongs to one or more synsets and each synset is made up of one or more words. Semantic relations like hypernymy and hyponymy exist between synsets, not between words! Consequently, there is no such thing like synonymy in Wordnet. If two words are synonymous the will share one or several synsets.
It is possible to access Wordnet is via the web interface: http://wordnetweb.princeton.edu/perl/webwn. There we can see e.g. the synsets of a word.

1. WordNet in Python

The NLTK package offers some easy methods to access WordNet. Before you use WordNet you have to run once the following code:

In [ ]:
```
import nltk

nltk.download("wordnet")
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package wordnet to /home/joji/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /home/joji/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[ ]:
```
True
```

How to access synsets:

In [ ]:
```
from nltk.corpus import wordnet as wn

# get synsets of a word
synsets = wn.synsets("rock")

for s in synsets:
    print(s)
print()

# use synset identifier directly
dog = wn.synset("dog.n.01")
print(dog.hypernyms())
```

```
print(dog.hyponyms())
print(dog.lemmas())  # ??
```

```
Synset('rock.n.01')
Synset('rock.n.02')
Synset('rock.n.03')
Synset('rock.n.04')
Synset('rock_candy.n.01')
Synset('rock_'n'_roll.n.01')
Synset('rock.n.07')
Synset('rock.v.01')
Synset('rock.v.02')

[Synset('canine.n.02'), Synset('domestic_animal.n.01')]
[Synset('basenji.n.01'), Synset('corgi.n.01'), Synset('cur.n.01'), Synset('dalmatian.n.02'), Synset('great_pyrenees.n.01'), Syn
set('griffon.n.02'), Synset('hunting_dog.n.01'), Synset('lapdog.n.01'), Synset('leonberg.n.01'), Synset('mexican_hairless.n.0
1'), Synset('newfoundland.n.01'), Synset('pooch.n.01'), Synset('poodle.n.01'), Synset('pug.n.01'), Synset('puppy.n.01'), Synset
('spitz.n.01'), Synset('toy_dog.n.01'), Synset('working_dog.n.01')]
[Lemma('dog.n.01.dog'), Lemma('dog.n.01.domestic_dog'), Lemma('dog.n.01.Canis_familiaris')]
```

An easy way to compute the similarity between two synsets is to measure the length of the path between the synsets in the WordNet hierarchy made up by the hypernym relations. The method path_simiarity returns 1/p where p is the length of the path between two synsets.

In [ ]:

```
ape = wn.synset("ape.n.01")
monkey = wn.synset("monkey.n.01")
zoo = wn.synset("zoo.n.01")

print( "Similarity between ape and monkey: ", ape.path_similarity(monkey))
print( "Similarity between ape and zoo: ", ape.path_similarity(zoo))
```

```
Similarity between ape and monkey:  0.3333333333333333
Similarity between ape and zoo:  0.07692307692307693
```

Wordnet is not completely connected. The path similarity method therefore assumes a fake root node that connect all parts. The path similarity has the problem that words are less similar if they are part of the hierarchy that is worked out in more detail. In general we would assume that the first divisions at the top of the hierarchy imply large semantic differences, while a division at a very deep position in the hierarchy makes only small semantic distinctions. Therefore some alternative measures have been defined, e.g. the Wu-Palmer similarity and the Leacock-Chodorow similarity (feel free to read up on those measures).

In [ ]:

```
print("Wu-Palmer similarity between ape and monkey: ", ape.wup_similarity(monkey))
print("Wu-Palmer similarity between ape and zoo: ", ape.wup_similarity(zoo))

print("Leacock Chodorow similarity between ape and monkey: ", ape.lch_similarity(monkey))
print("Leacock Chodorow similarity between ape and zoo: ", ape.lch_similarity(zoo))
```

```
Wu-Palmer similarity between ape and monkey:  0.9230769230769231
Wu-Palmer similarity between ape and zoo:  0.4
Leacock Chodorow similarity between ape and monkey:  2.538973871058276
Leacock Chodorow similarity between ape and zoo:  1.072636802264849
```

Both measures give higher weight to distances between nodes that are closer to the root. However, the distance to the root is also a design decision and a number of measures try to include other information sources as well. E.g. the similarity measures of Resnik and Lin include the frequency of words in a corpus as well.

2. Exercise:

1. Read in the email dataset (see exercise 3). You may copy some of the code from that notebook.
2. Let us investigate the coverage of this data in Wordnet:
   - Count the unique words (types) in the data and store them in a list.
   - How many of those items have synsets in Wordnet? (calculate a percentage value)
   - What is the average number of synsets per type?
3. Not all words have lexical meaning. We can filter certain word classes. Apply POS-tagging (for example https://www.nltk.org/book/ch05.html) to extract only nouns (just NN - not proper nouns NNP). Check the coverage in Wordnet for these nouns. How many have synsets in Wordnet? (calculate a percentage value)
4. Experiments with the similarity of words:
   - Choose 10 out of the 50 most frequent nouns from the data set (they all should have at least one synset in Wordnet).
   - Now compute for each of the 10 words the Wu-Palmer or Leacock-Chodorow similarity to each of the other 9 words (you may use the first synset for each word for this calculation when words have multiple synsets). You might want to display the resulting numbers in a table. Which words are most similar to each other?
   - Check for all sentences which contain the word 'Obama': How often does each of the 10 words you selected occur in these sentences? Have words with similar meaning also similar co-occurrence counts with 'Obama'?

In [ ]:

```
# 0

# read the file
texts = open('emails-body.txt').read().split('<cmail>\n')
# str_texts = ",".join(texts)
# print(type(str_texts))
```

```
# print(str_texts)
print(f'number of email bodies: {len(texts)}')
```

```
number of email bodies: 6741
```

In [ ]:

```
# 1.1
# Counting words(types)

from somajo import SoMaJo

somajo_tokenizer = SoMaJo(language="en_PTB", split_camel_case=True)

# this might take a minute
data_tok = []
for sentence in somajo_tokenizer.tokenize_text(texts):
    data_tok.append([token.text for token in sentence])
# print(len(data_tok))
# print(data_tok[200])

# print(data_tok)
```

In [ ]:

```
# count words and their frequencies
from collections import Counter

sentences = data_tok
# print(sentences)

words = Counter(word for sentence in sentences for word in sentence)
# Note: "words" now contains a mapping of words to their frequencies.

# total number of types in the corpus
print(f'Total number of types (unique words): {len(words)}')
```

```
Total number of types (unique words): 37340
```

In [ ]:

```
# 1.2
# how many in 'words' have synsets in wordnet

from nltk.corpus import wordnet as wn

wordList = list(words)

cnt = 0
totalSynsets = []
synsetLengths = []
for w in wordList:
```

```python
    synsets = wn.synsets(w)
    # print('synsets is {} and len is {}'.format(synsets, len(synsets)))
    if len(synsets) > 0:
        synsetLengths.append(len(synsets))
        cnt += 1
# print(cnt)
# print(sum(synsetLengths)/len(wordList))
# print(len(synsetLengths))
# print(len(wordList))

print('Percentage of words having synsets', 100*(cnt/len(wordList)))
```

```
Percentage of words having synsets 66.24799143010178
```

In [ ]:

```python
# 1.3
# avg number of synsets per word
import numpy as np
print('The average number of synsets per word is',sum(synsetLengths)/len(wordList))
```

```
The average number of synsets per word is 3.0951526513122656
```

In [ ]:

```python
# 2
# Not all words have lexical meaning. We can filter certain word classes. Apply POS-tagging (for example https://www.nltk.org/b
#  (just NN - not proper nouns NNP). Check the coverage in Wordnet for these nouns. How many have synsets in Wordnet? (calculat
import nltk
from nltk import grammar, parse
from nltk.tokenize import word_tokenize

data_tok = [[token.lower() for token in sentence] for sentence in data_tok]
flat_list = [item for sublist in data_tok for item in sublist]
# print(flat_list)

# print(type(data_tok[1]))
tagged_tokens = nltk.pos_tag(flat_list)
# print(tagged_tokens)

NN_lst = []
for ind in range(len(tagged_tokens)):
    if tagged_tokens[ind][1] == 'NN':
        NN_lst.append(tagged_tokens[ind][0])
# print('nouns lst', NN_lst)

ncnt = 0
NN_synsets = []
for n in NN_lst:
    synsets = wn.synsets(n)
    if len(synsets) > 0:
```

```
        NN_synsets.append(n)
        ncnt += 1

# print(ncnt)
print("Wordnet coverage for nouns in % is: ", 100*(ncnt/len(NN_lst)))
```

```
Wordnet coverage for nouns in % is:  81.92861159813424
```

In [ ]:
```
# 3.1
# Instead of selecting randomly from top 50, I am taking the top 10 most common nouns that have at least 1 synsets
import collections
counter_NN_synsets = collections.Counter(NN_synsets)
# print(counter_NN_lst)
mostCommon = counter_NN_synsets.most_common(50)

plain_lst = [key for key, _ in mostCommon]
print('Top 50 most common',plain_lst)

plain_lst = ['president', 'staff', 'department', 'government', 'clinton', 'washington', 'work', 'country','minister', 'administ
print()
print('Selected nouns: ', plain_lst)
```

```
Top 50 most common ['i', 'pm', 'state', 'secretary', 'office', 'president', 'time', 'house', 'department', 'government', 'clint
on', 'today', 'meeting', 'policy', 'security', 'h', 'party', 'world', 'room', 'way', 'call', 'year', 'minister', 'administratio
n', 'tomorrow', 're', 'case', 'bill', 'day', 'week', 'staff', 'israel', 'washington', 'conference', 'part', 'work', 'health',
'country', 'agreement', 'information', 'group', 'w', 'support', 'percent', 'peace', 'speech', 'war', 'date', 'benghazi', 'afgha
nistan']

Selected nouns:  ['president', 'staff', 'department', 'government', 'clinton', 'washington', 'work', 'country', 'minister', 'ad
ministration']
```

In [ ]:
```
# 3.2
# Now compute for each of the 10 words the Wu-Palmer or Leacock-Chodorow similarity to each of the other 9 words
# (you may use the first synset for each word for this calculation when words have multiple synsets).
# You might want to display the resulting numbers in a table. Which words are most similar to each other?
print('words similar to eachother as seen in this section are STAFF - DEPARTMENT and DEPARTMENT - GOVERNMENT')
print()
from nltk.corpus import wordnet as wn
for w1 in plain_lst:
    # print('i am here')
    for w2 in plain_lst:
        if w1 != w2:
            synsets1 = wn.synsets(w1)
            word_a = synsets1[0]
            synsets2 = wn.synsets(w2)
            word_b = synsets2[0]
            print("Leacock Chodorow similarity between {} and {}: {}".format(w1, w2, word_a.lch_similarity(word_b)))
```

words similar to eachother as seen in this section are STAFF - DEPARTMENT and DEPARTMENT - GOVERNMENT

```
Leacock Chodorow similarity between president and staff: 0.8649974374866046
Leacock Chodorow similarity between president and department: 0.7472144018302211
Leacock Chodorow similarity between president and government: 0.8649974374866046
Leacock Chodorow similarity between president and clinton: 1.3350010667323402
Leacock Chodorow similarity between president and washington: 0.7472144018302211
Leacock Chodorow similarity between president and work: 0.8649974374866046
Leacock Chodorow similarity between president and country: 0.8043728156701697
Leacock Chodorow similarity between president and minister: 1.4403615823901665
Leacock Chodorow similarity between president and administration: 0.8043728156701697
Leacock Chodorow similarity between staff and president: 0.8649974374866046
Leacock Chodorow similarity between staff and department: 1.6916760106710724
Leacock Chodorow similarity between staff and government: 2.0281482472922856
Leacock Chodorow similarity between staff and clinton: 0.9295359586241757
Leacock Chodorow similarity between staff and washington: 0.8043728156701697
Leacock Chodorow similarity between staff and work: 1.2396908869280152
Leacock Chodorow similarity between staff and country: 1.845826690498331
Leacock Chodorow similarity between staff and minister: 0.9985288301111273
Leacock Chodorow similarity between staff and administration: 1.1526795099383855
Leacock Chodorow similarity between department and president: 0.7472144018302211
Leacock Chodorow similarity between department and staff: 1.6916760106710724
Leacock Chodorow similarity between department and government: 1.6916760106710724
Leacock Chodorow similarity between department and clinton: 0.8043728156701697
Leacock Chodorow similarity between department and washington: 0.6931471805599453
Leacock Chodorow similarity between department and work: 1.072636802264849
Leacock Chodorow similarity between department and country: 1.845826690498331
Leacock Chodorow similarity between department and minister: 0.8649974374866046
Leacock Chodorow similarity between department and administration: 0.9985288301111273
Leacock Chodorow similarity between government and president: 0.8649974374866046
Leacock Chodorow similarity between government and staff: 2.0281482472922856
Leacock Chodorow similarity between government and department: 1.6916760106710724
Leacock Chodorow similarity between government and clinton: 0.9295359586241757
Leacock Chodorow similarity between government and washington: 0.8043728156701697
Leacock Chodorow similarity between government and work: 1.2396908869280152
Leacock Chodorow similarity between government and country: 1.845826690498331
Leacock Chodorow similarity between government and minister: 0.9985288301111273
Leacock Chodorow similarity between government and administration: 1.1526795099383855
Leacock Chodorow similarity between clinton and president: 1.3350010667323402
Leacock Chodorow similarity between clinton and staff: 0.9295359586241757
Leacock Chodorow similarity between clinton and department: 0.8043728156701697
Leacock Chodorow similarity between clinton and government: 0.9295359586241757
Leacock Chodorow similarity between clinton and washington: 0.8043728156701697
Leacock Chodorow similarity between clinton and work: 0.9295359586241757
Leacock Chodorow similarity between clinton and country: 0.8649974374866046
Leacock Chodorow similarity between clinton and minister: 1.55814461804655
```

```
Leacock Chodorow similarity between clinton and administration: 0.8649974374866046
Leacock Chodorow similarity between washington and president: 0.7472144018302211
Leacock Chodorow similarity between washington and staff: 0.8043728156701697
Leacock Chodorow similarity between washington and department: 0.6931471805599453
Leacock Chodorow similarity between washington and government: 0.8043728156701697
Leacock Chodorow similarity between washington and clinton: 0.8043728156701697
Leacock Chodorow similarity between washington and work: 0.8043728156701697
Leacock Chodorow similarity between washington and country: 0.7472144018302211
Leacock Chodorow similarity between washington and minister: 0.8649974374866046
Leacock Chodorow similarity between washington and administration: 0.7472144018302211
Leacock Chodorow similarity between work and president: 0.8649974374866046
Leacock Chodorow similarity between work and staff: 1.2396908869280152
Leacock Chodorow similarity between work and department: 1.072636802264849
Leacock Chodorow similarity between work and government: 1.2396908869280152
Leacock Chodorow similarity between work and clinton: 0.9295359586241757
Leacock Chodorow similarity between work and washington: 0.8043728156701697
Leacock Chodorow similarity between work and country: 1.1526795099383855
Leacock Chodorow similarity between work and minister: 0.9985288301111273
Leacock Chodorow similarity between work and administration: 1.6916760106710724
Leacock Chodorow similarity between country and president: 0.8043728156701697
Leacock Chodorow similarity between country and staff: 1.845826690498331
Leacock Chodorow similarity between country and department: 1.845826690498331
Leacock Chodorow similarity between country and government: 1.845826690498331
Leacock Chodorow similarity between country and clinton: 0.8649974374866046
Leacock Chodorow similarity between country and washington: 0.7472144018302211
Leacock Chodorow similarity between country and work: 1.1526795099383855
Leacock Chodorow similarity between country and minister: 0.9295359586241757
Leacock Chodorow similarity between country and administration: 1.072636802264849
Leacock Chodorow similarity between minister and president: 1.4403615823901665
Leacock Chodorow similarity between minister and staff: 0.9985288301111273
Leacock Chodorow similarity between minister and department: 0.8649974374866046
Leacock Chodorow similarity between minister and government: 0.9985288301111273
Leacock Chodorow similarity between minister and clinton: 1.55814461804655
Leacock Chodorow similarity between minister and washington: 0.8649974374866046
Leacock Chodorow similarity between minister and work: 0.9985288301111273
Leacock Chodorow similarity between minister and country: 0.9295359586241757
Leacock Chodorow similarity between minister and administration: 0.9295359586241757
Leacock Chodorow similarity between administration and president: 0.8043728156701697
Leacock Chodorow similarity between administration and staff: 1.1526795099383855
Leacock Chodorow similarity between administration and department: 0.9985288301111273
Leacock Chodorow similarity between administration and government: 1.1526795099383855
Leacock Chodorow similarity between administration and clinton: 0.8649974374866046
Leacock Chodorow similarity between administration and washington: 0.7472144018302211
Leacock Chodorow similarity between administration and work: 1.6916760106710724
Leacock Chodorow similarity between administration and country: 1.072636802264849
Leacock Chodorow similarity between administration and minister: 0.9295359586241757
```

In [ ]:

```
# 3.3
# Check for all sentences which contain the word 'Obama': How often does each of the 10 words you
# selected occur in these sentences? Have words with similar meaning also similar co-occurrence counts with 'Obama'?

# print(sentences[:10])
text_str = ''.join(texts)
# print(text_str)
from nltk.tokenize import sent_tokenize, word_tokenize
sentences_1 = sent_tokenize(text_str)
# print(sentences_1)
# print(type(sentences_1))
# print(len(sentences_1))
```

In [ ]:

```
obama_lst = []
for elems in sentences_1:
    if 'Obama' in elems:
        obama_lst.append(elems)
# print(len(obama_lst))
```

In [ ]:

```
occurence_counts = [0]*10
for p in range(len(plain_lst)):
    for occur in obama_lst:
        if plain_lst[p] in occur:
            occurence_counts[p] += 1
# print(occurence_counts, 'for selected words :', plain_lst)
print('selected words: ', plain_lst, 'respective their counts are: ', occurence_counts)
```

```
selected words:  ['president', 'staff', 'department', 'government', 'clinton', 'washington', 'work', 'country', 'minister', 'ad
ministration'] respective their counts are:  [96, 19, 3, 30, 4, 1, 38, 20, 8, 118]
```

In [ ]:

```

```