

NLP Assignment - Words and Friends

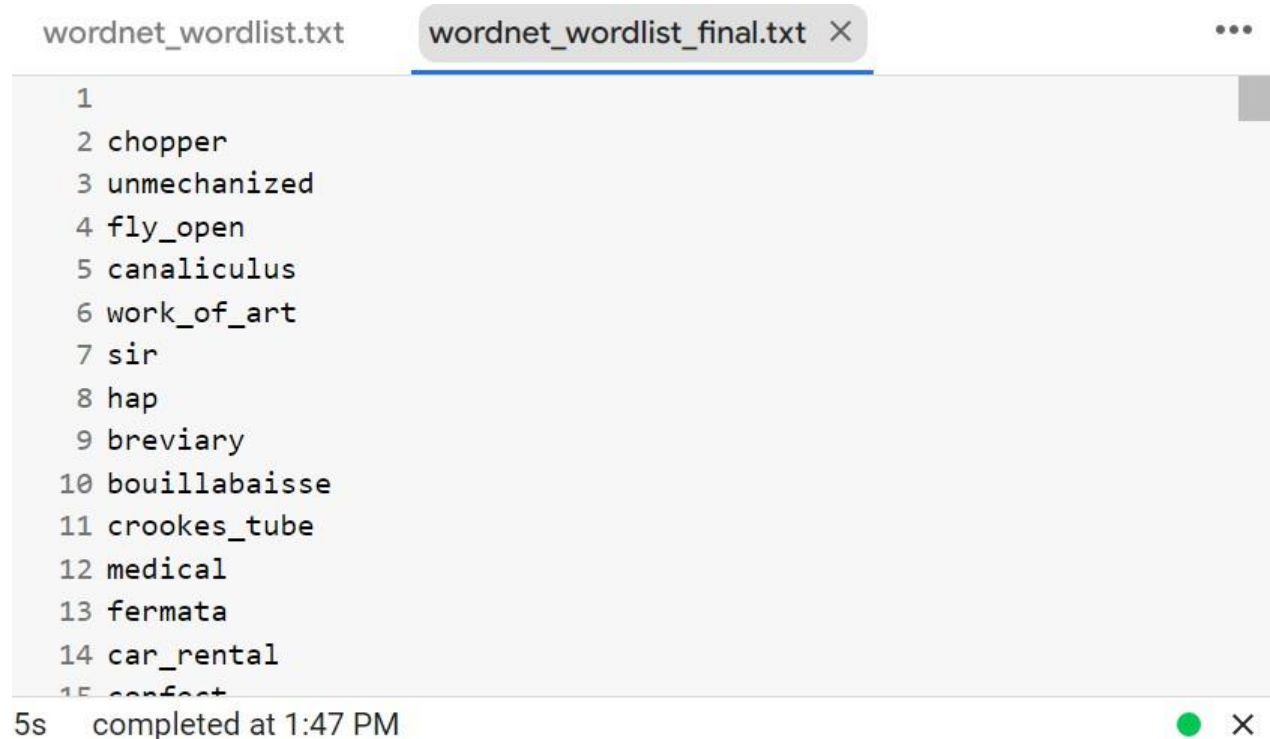
Part A:

Creating a wordlist file from all the words in wordnet:

```
from nltk.corpus import wordnet
#Extracting all the words from wordnet.
with open("wordnet_wordlist.txt","w") as f:
    for words in list(wordnet.all_synsets()):
        f.write(words.name()[:-5])
        f.write("\n")

f = open("wordnet_wordlist.txt")
with open("wordnet_wordlist_final.txt", "w") as f2:
    uniquelines = set(f.read().split("\n"))
    f2.write("".join([line + "\n" for line in uniquelines]))
f.close()
```

Output:



```
wordnet_wordlist.txt  wordnet_wordlist_final.txt X ...
```

```
1
2 chopper
3 unmechanized
4 fly_open
5 canaliculus
6 work_of_art
7 sir
8 hap
9 breviary
10 bouillabaisse
11 crookes_tube
12 medical
13 fermata
14 car_rental
15 confect
```

5s completed at 1:47 PM

Given a word use WordNet to:

Get distance 0 words.

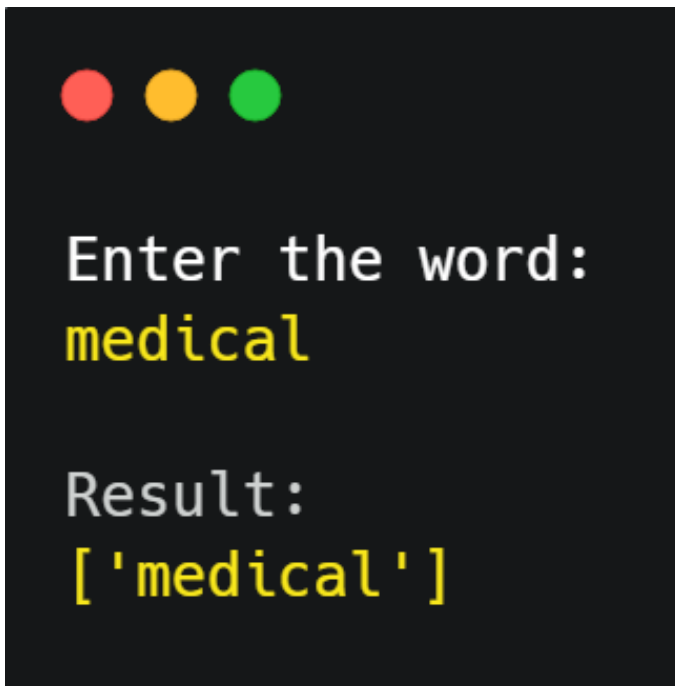
Code:

```
#Finding words at a 0 distance.
f = open("./wordnet_wordlist_final.txt", "r")
words = f.readlines()
words = [w[:-1] for w in words]

def get_0_distance_words(word):
    result = []
    for i in words:
        if nltk.edit_distance(word, i) == 0:
            result.append(i)
    return result

inp = input("Enter the word:\n")
print()
print("Result:")
print(get_0_distance_words(inp))
```

Output:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. The text inside the terminal is as follows:

```
Enter the word:
medical

Result:
['medical']
```

get distance 'n' words where 'n' is between 1 to 4 with all the semantic relations.

Code:

```
#Finding words at a particular distance.
f = open("./wordnet_wordlist_final.txt", "r")
words = f.readlines()
words = [w[:-1] for w in words]

def get_n_distance_words(word, dist):
    result = []
    for i in words:
        if nltk.edit_distance(word, i) == int(dist):
            result.append(i)
    return result

inp = input("Enter the word:")
d = input("Enter the distance:")
print(get_n_distance_words(inp, d))
```

Output:



Enter the word:

love

Enter the distance:

2

Result:

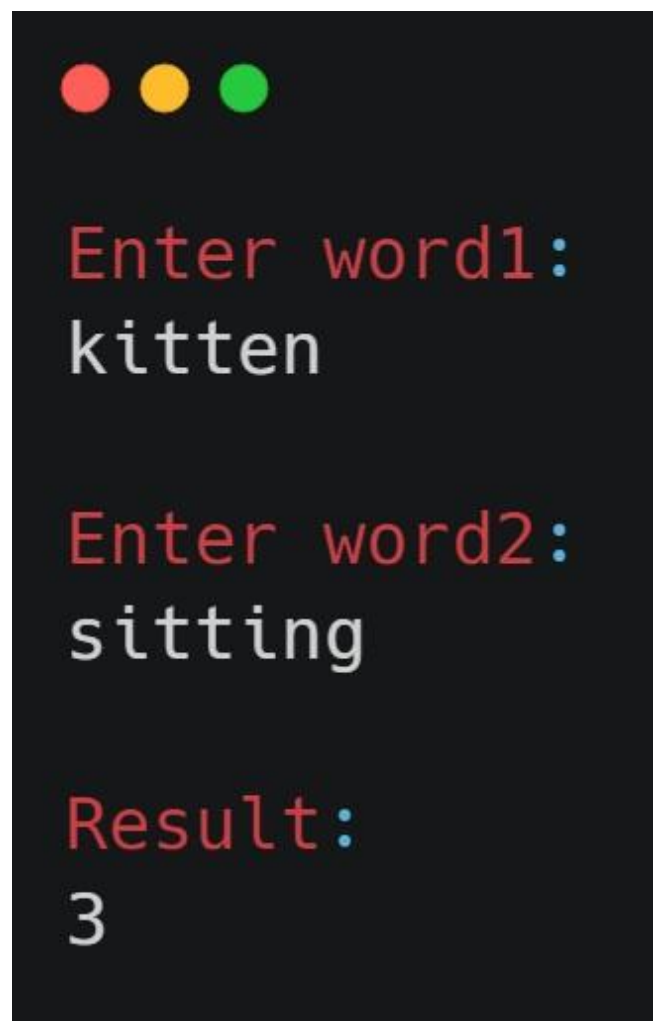
```
['slave', 'robe', 'pole', 'olive', 'alcove', 'lyre', 'mole', 'one', 'lie',  
'doze', 'gone', 'globe', 'moved', 'core', 'bore', 'doge', 'vole', 'luce',  
'oven', 'tome', 'woven', 'level', 'loeb', 'louver', 'yoke', 'loon', 'load',  
'sole', 'stove', 'loti', 'dose', 'lodz', 'woe', 'aloe', 'pome', 'coue', 'hovel',  
'clover', 'bone', 'late', 'more', 'lame', 'liver', 'lolo', 'dope', 'loaf',  
'lime', 'wave', 'hoe', 'none', 'five', 'low', 'poke', 'luge', 'leone', 'lace',  
'lovage', 'tone', 'clive', 'lake', 'loser', 'pose', 'dole', 'loos', 'rave',  
'hope', 'dover', 'lane', 'nome', 'dote', 'some', 'lever', 'foe', 'moke', 'loft',  
'loren', 'mope', 'locke', 'role', 'lower', 'doe', 'bode', 'prove', 'loner',  
'lox', 'louse', 'poe', 'line', 'slope', 'mover', 'alone', 'lovell', 'hive',  
'loin', 'loire', 'code', 'lovely', 'zone', 'rome', 'done', 'pope', 'alive',  
'mode', 'rover', 'loewe', 'cave', 'save', 'hover', 'lure', 'hole', 'clone',  
'log', 'like', 'cone', 'loud', 'lord', 'lorre', 'loupe', 'loose', 'lava',  
'plover', 'rope', 'cover', 'shove', 'loan', 'nose', 'gloved', 'coven', 'ploc',  
'vote', 'dome', 'ode', 'joke', 'kobe', 'jive', 'loop', 'over', 'life', 'soave',  
'lodge', 'cope', 'lory', 'solve', 'toke', 'livy', 'elope', 'close', 'luke',  
'lovoa', 'sore', 'fore', 'bole', 'bose', 'drove', 'tole', 'lute', 'wive',  
'lout', 'cloze', 'toe', 'lock', 'roe', 'sone', 'lobed', 'covet', 'lot', 'lost',  
'loom', 'home', 'grove', 'lee', 'rote', 'neve', 'have', 'ore', 'fovea', 'howe',  
'yore', 'gore', 'sloe', 'loki', 'rose', 'long', 'hovea', 'nave', 'lev', 'dive',  
'loot', 'cloven', 'note', 'loir', 'loch', 'node', 'loss', 'coke', 'novel',  
'laver', 'movie', 'hone', 'louvre', 'owe', 'eve', 'lota', 'pore', 'come',  
'give', 'hose', 'logo', 'above', 'levy', 'lye', 'covey', 'look', 'loam', 'nova',  
'levee', 'pave', 'cote', 'leave', 'lob']
```

Given two words give the minimum distance between them with path.

Code:

```
from nltk import edit_distance
word1 = input("Enter word1:\n")
word2 = input("Enter word2:\n")
print("\nResult:")
print(edit_distance(word1, word2))
```

Output:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. The text is displayed in a monospaced font, with prompts in red and user input/output in white. The prompts are "Enter word1:", "Enter word2:", and "Result:". The user input is "kitten" for word1 and "sitting" for word2. The final output is "3".

```
Enter word1:
kitten

Enter word2:
sitting

Result:
3
```

Part B

Given a word, w search wikipedia to find the words that are friend with w.

```
import requests
from nltk import edit_distance
from bs4 import BeautifulSoup
from nltk import word_tokenize
from nltk.corpus import wordnet
nltk.download('punkt')
nltk.download('wordnet')

given_word = input("Enter a word:\n")
url = input("Enter a wikipedia URL:\n")
page = requests.get(url).content
soup = BeautifulSoup(page)
text = " ".join([sentences.text for sentences in soup.find_all('p')])
words_list = word_tokenize(text)
# print(text)
# print(words)

wordnet_similar_words = []
syns = wordnet.synsets(given_word)
for i in syns:
    for j in i.lemmas():
        wordnet_similar_words.append(str(j.name()))

result = []
for word in words_list:
    if word in wordnet_similar_words:
        result.append(word)
if len(result) == 0:
    print("No matching words found !")
else:
    print("All similar words:")
    print(result)
    print()
    print("Unique words:")
    print(set(result))
```

Output:



Enter a word:

education

Enter a wikipedia URL:

https://en.wikipedia.org/wiki/Primary_school

All similar words:

```
['education', 'Education', 'education', 'education', 'education', 'education',  
'education', 'teaching', 'teaching', 'education', 'education', 'education',  
'teaching', 'education', 'education', 'education', 'education', 'education',  
'education', 'education', 'education', 'education', 'education', 'education',  
'education', 'education', 'education', 'teaching', 'education']
```

Unique words:

```
{'teaching', 'education', 'Education'}
```

Part C

Given a word, w search search wiktionary to find the words that are friend with w.

Code:

```
from nltk import word_tokenize
given_word = input("Enter a word:\n")

with open("/content/enwiktionary-20220401-pages-articles-multistream-
index.txt") as f:
    text = f.readlines()

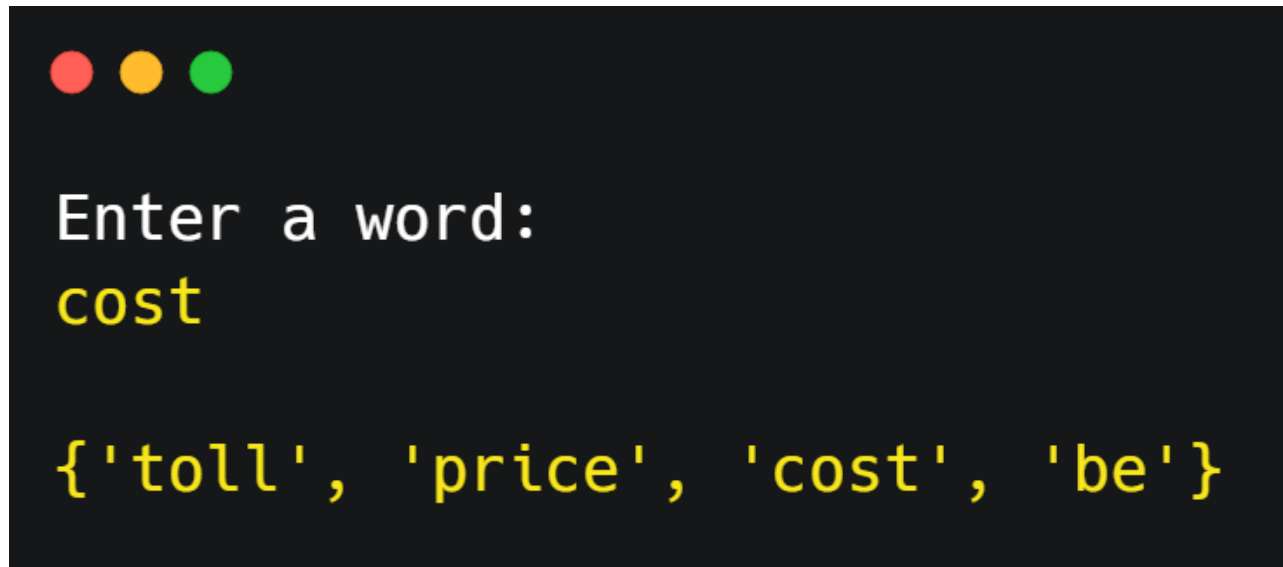
i = 0
for words in text:
    try:
        text[i] = words.split(":")[-1]
        i += 1
    except:
        continue

i = 0
for words in text:
    try:
        text[i] = re.sub("\\n", "", words)
        i += 1
    except:
        continue
# df = pd.DataFrame(text)

wordnet_similar_words = []
syns = wordnet.synsets(given_word)
for i in syns:
    for j in i.lemmas():
        wordnet_similar_words.append(str(j.name()))

result = []
for word in text:
    if word in wordnet_similar_words:
        result.append(word)
if len(result) == 0:
    print("No matching words found !")
else:
    print(set(result))
```


Output:



```
Enter a word:  
cost  
  
{'toll', 'price', 'cost', 'be'}
```