Vidyavardhini's College of Engineering and Technology, Vasai

Department of Computer Science & Engineering (Data Science)

AY: 2025-26

Class:	**BE-CSE(DS)**	**Semester:**	**VII**
Course Code:	**CSDOL7011**	**Course Name:**	**NLP Lab**

Name of Student:	**Sahil Salunke**
Roll No. :	**45**
Experiment No.:	**7**
Title of the Experiment:	**Calculating Semantic Similarity and Performing Word Sense Disambiguation using WordNet**
Date of Performance:	
Date of Submission:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty :

Signature :

Date :

Aim: To use WordNet for computing semantic similarity between words and performing basic word sense disambiguation.

Objective: To compute semantic similarity and perform word sense disambiguation using WordNet.

Tools Required:

1. Python (Jupyter Notebook or Google Colab)
2. nltk

Procedure:

1. Import required libraries and download WordNet:
    - a. import nltk
    - b. from nltk.corpus import wordnet as wn
    - c. nltk.download('wordnet')
    - d. nltk.download('omw-1.4')

2. Compute WordNet-based similarity:
    - a. Choose two words (e.g., car and automobile)
    - b. Fetch their synsets:
        - i. syn1 = wn.synsets('car')[0]
        - ii. syn2 = wn.synsets('automobile')[0]
        - iii. similarity = syn1.wup_similarity(syn2)
        - iv. print(f"Similarity = {similarity}")

3. Compare similarity of different word pairs:

Try synonyms, hypernyms, unrelated words and observe scores.

4. Word Sense Disambiguation (WSD):

Use simplified Lesk algorithm:

a. from nltk.wsd import lesk
b. from nltk.tokenize import word_tokenize
c. sentence = "The bank can guarantee deposits will eventually cover future tuition costs."
d. ambiguous = "bank"
e. context = word_tokenize(sentence)
f. sense = lesk(context, ambiguous)
g. print(f"Best sense for '{ambiguous}': {sense.definition()}")

Description of the Experiment:

Students use WordNet, a lexical database, to analyze word meaning and similarity. This experiment introduces the concept of semantic relationships like synonymy, hypernymy, and helps distinguish between different senses of the same word depending on context (disambiguation).

Detailed Description of the NLP Technique:

1. WordNet:

WordNet is a large lexical database of English where:

Nouns, verbs, adjectives, and adverbs are grouped into synsets (sets of synonyms).

Synsets are interlinked by semantic relations such as:

Hypernyms (generalization, e.g., animal → dog)

Hyponyms (specialization, e.g., dog → pug)

Meronyms (part-whole relationships)

2. Semantic Similarity:

WordNet provides multiple similarity measures:

Path Similarity: Based on shortest path in hierarchy.

Wu-Palmer Similarity: Based on the depth of the least common subsumer (used in this experiment).

These scores range from 0 (no similarity) to 1 (identical meanings).

3. Word Sense Disambiguation (WSD):

WSD is the task of determining which sense of a word is activated by its context in a sentence.

Lesk Algorithm: Disambiguates word sense by comparing dictionary definitions (glosses) of each sense with the words in the surrounding context.

4. Applications of Semantic Analysis:

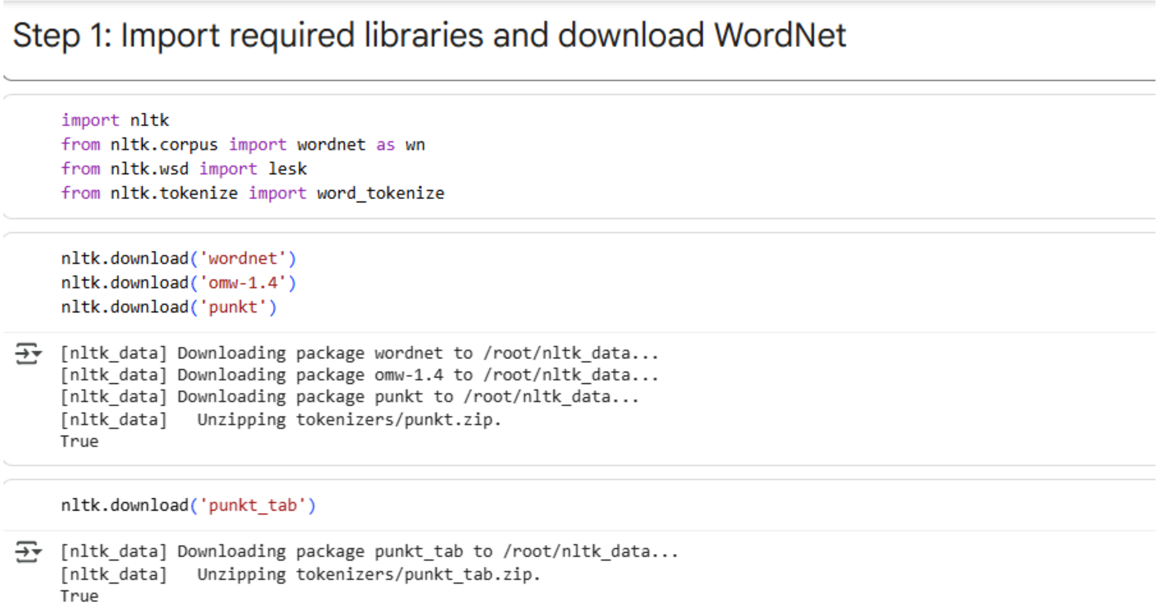
Question Answering Systems

Chatbots and Dialogue Systems

Machine Translation

Information Retrieval

OUTPUT:



Step 1: Import required libraries and download WordNet

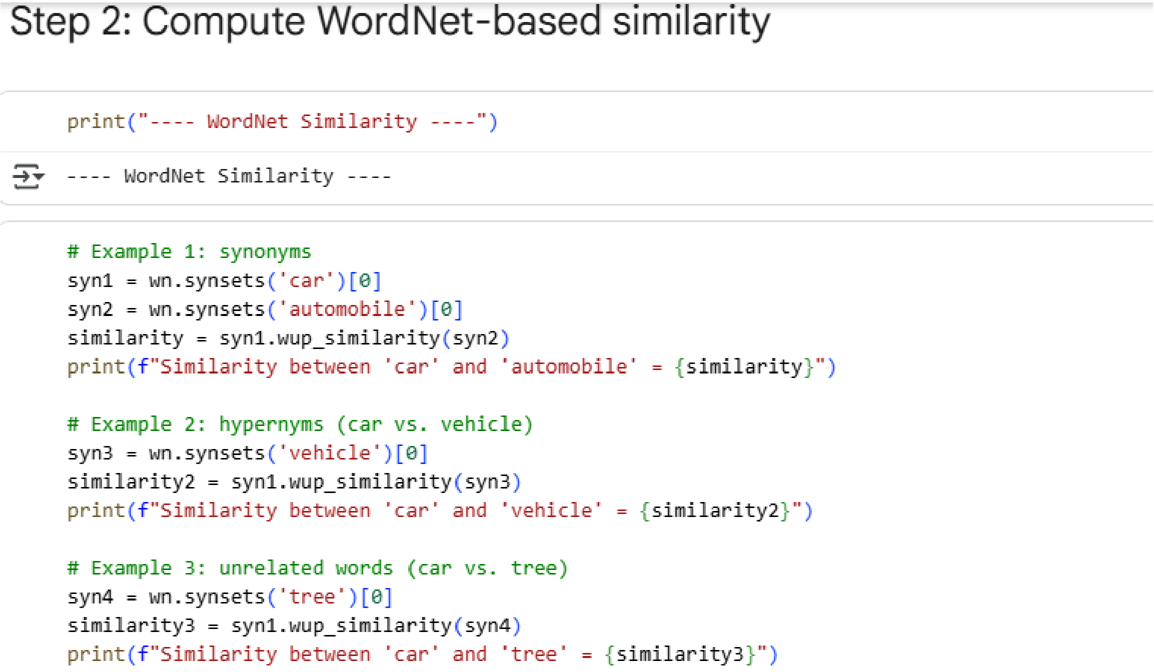
```
import nltk
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.tokenize import word_tokenize
```

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

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

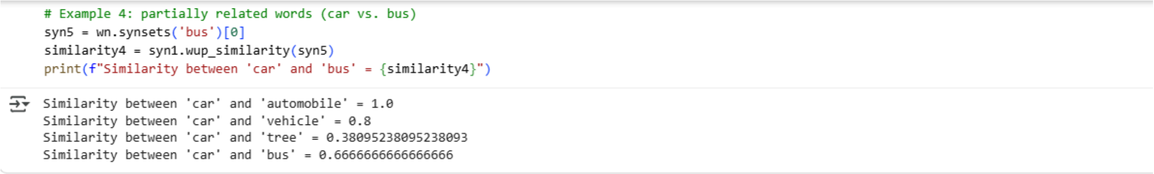


Step 2: Compute WordNet-based similarity

```
print("---- WordNet Similarity ----")
```

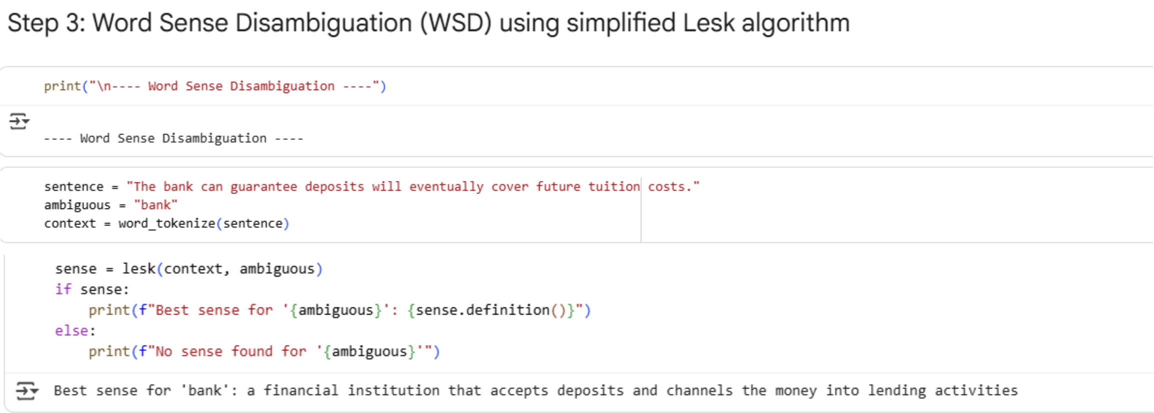
```
---- WordNet Similarity ----
```

```
# Example 1: synonyms
syn1 = wn.synsets('car')[0]
syn2 = wn.synsets('automobile')[0]
similarity = syn1.wup_similarity(syn2)
print(f"Similarity between 'car' and 'automobile' = {similarity}")

# Example 2: hypernyms (car vs. vehicle)
syn3 = wn.synsets('vehicle')[0]
similarity2 = syn1.wup_similarity(syn3)
print(f"Similarity between 'car' and 'vehicle' = {similarity2}")

# Example 3: unrelated words (car vs. tree)
syn4 = wn.synsets('tree')[0]
similarity3 = syn1.wup_similarity(syn4)
print(f"Similarity between 'car' and 'tree' = {similarity3}")
```



```
# Example 4: partially related words (car vs. bus)
syn5 = wn.synsets('bus')[0]
similarity4 = syn1.wup_similarity(syn5)
print(f"Similarity between 'car' and 'bus' = {similarity4}")
```

```
Similarity between 'car' and 'automobile' = 1.0
Similarity between 'car' and 'vehicle' = 0.8
Similarity between 'car' and 'tree' = 0.38095238095238093
Similarity between 'car' and 'bus' = 0.6666666666666666
```



Step 3: Word Sense Disambiguation (WSD) using simplified Lesk algorithm

```
print("\n---- Word Sense Disambiguation ----")
```

```

---- Word Sense Disambiguation ----
```

```
sentence = "The bank can guarantee deposits will eventually cover future tuition costs."
ambiguous = "bank"
context = word_tokenize(sentence)
```

```
sense = lesk(context, ambiguous)
if sense:
    print(f"Best sense for '{ambiguous}': {sense.definition()}")
else:
    print(f"No sense found for '{ambiguous}'")
```

```
Best sense for 'bank': a financial institution that accepts deposits and channels the money into lending activities
```

Conclusion:

- WordNet similarity works well for word-level lexical comparisons, especially when words are synonyms or belong to the same semantic category.
- The Simplified Lesk algorithm successfully disambiguates polysemous words like *bank* by leveraging contextual clues.
- Together, they demonstrate the usefulness of lexical resources (WordNet) in semantic similarity tasks and WSD applications such as search engines, chatbots, and information retrieval.