Name:Rishi Patel
Batch:CS8
Roll No:70
PRN:202401120070

20 Problem Statements on WordNet Dataset

```
# 1. Top 10 synsets with highest number of synonyms

top_synsets = sorted(wn.all_synsets(), key=lambda s:
len(s.lemma_names()), reverse=True)[:10]
print("\n1. Top 10 Synsets with Most
Synonyms:") for syn in top_synsets:
    print(syn.name(),

syn.lemma_names()) # 2. Word with

maximum different senses

lemmas = wn.all_lemma_names()
sense_count = {lemma: len(wn.synsets(lemma)) for lemma in
lemmas} most_senses = max(sense_count, key=sense_count.get)
print("\n2. Word with Most Senses:", most_senses,
sense_count[most_senses])

# 3. All synonyms of a word (e.g., happy)

word = 'happy'
synonyms = set()
for syn in
    wn.synsets(word): for
    lemma in syn.lemmas():
        synonyms.add(lemma.name())
print("\n3. Synonyms of 'happy':",
synonyms)

# 4. Words having both noun and verb senses

both_nv = [lemma for lemma in lemmas if
wn.synsets(lemma, pos=wn.NOUN) and
wn.synsets(lemma, pos=wn.VERB)] print("\n4. Words with
both Noun and Verb senses:", both_nv[:10])
```

```
# 5. Most common hypernyms

hypernym_counter = {}
for syn in wn.all_synsets('n'):
    for hyper in
    syn.hypernyms():
        hypernym_counter[hyper.name()] =
        hypernym_counter.get(hyper.name(),
0) + 1
common_hypernyms = sorted(hypernym_counter.items(),
key=lambda x: x[1], reverse=True)[:10]
print("\n5. Most Common Hypernyms:",

common_hypernyms) # 6. All pairs of antonyms

antonym_pairs = []
for syn in wn.all_synsets():
    for lemma in
    syn.lemmas():
        if lemma.antonyms():
            antonym_pairs.append((lemma.name(),
lemma.antonyms()[0].name())) print("\n6. Antonym Pairs:",
antonym_pairs[:10])

# 7. Words with most antonyms

antonym_count = {}
for lemma in
lemmas:
    antonyms = []
    for syn in
        wn.synsets(lemma): for l
        in syn.lemmas():
            antonyms += l.antonyms()
    antonym_count[lemma] = len(antonyms)
most_antonyms = max(antonym_count,
key=antonym_count.get) print("\n7. Word with Most
Antonyms:", most_antonyms)

# 8. Synset counts per part of speech

pos_counts = {pos: len(list(wn.all_synsets(pos))) for pos in [wn.NOUN,
wn.VERB, wn.ADJ, wn.ADV]}
```

```
print("\n8. Synsets per Part of Speech:",

pos_counts) # 9. Synsets with 5+ hyponyms

large_synsets = []
for syn in wn.all_synsets():
    if len(syn.hyponyms()) >= 5:
        large_synsets.append(syn.name())
print("\n9. Synsets with 5+ Hyponyms:", large_synsets[:10])
```

```
# 10. All hyponyms of a word (e.g., vehicle)

vehicle_synsets = wn.synsets('vehicle', pos=wn.NOUN)
vehicle_hyponyms = set()
for syn in vehicle_synsets:
    for hypo in syn.hyponyms():
        vehicle_hyponyms.add(hypo.name())
print("\n10. Hyponyms of 'vehicle':",

vehicle_hyponyms) # 11. Average number of

synonyms per synset

syn_counts = [len(syn.lemma_names()) for syn in wn.all_synsets()]
avg_synonyms = np.mean(syn_counts)
print("\n11. Average Synonyms per Synset:",

avg_synonyms) # 12. Words that are adjectives and

adverbs

adj_adv = [lemma for lemma in lemmas if wn.synsets(lemma,
pos=wn.ADJ) and wn.synsets(lemma, pos=wn.ADV)]
print("\n12. Words both Adjective and Adverb:",

adj_adv[:10]) # 13. Maximum depth in noun hierarchy

max_depth = max(syn.min_depth() for syn in
wn.all_synsets('n')) print("\n13. Maximum Depth in Noun
Hierarchy:", max_depth)

# 14. Direct hypernyms of a word (e.g., dog)

dog_synsets = wn.synsets('dog',
pos=wn.NOUN) hypernyms_dog = []
for syn in dog_synsets:
    hypernyms_dog +=
    syn.hypernyms()
print("\n14. Direct Hypernyms of 'dog':", [h.name() for h in
hypernyms_dog])
```

```
# 15. Words with multiple hypernyms

multi_hypernym_words =
[] for syn in
wn.all_synsets('n'):
    if len(syn.hypernyms()) > 1:
        multi_hypernym_words.append(syn.na
        me())
print("\n15. Words with Multiple Hypernyms:",
multi_hypernym_words[:10])




# 16. Synsets with no hypernyms (root concepts)

root_synsets = [syn.name() for syn in wn.all_synsets('n') if not
syn.hypernyms()]
print("\n16. Root Synsets:", root_synsets[:10])

# 17. Top 10 words closely related to 'car' (path similarity)

car_synsets =
wn.synsets('car') similarities
= {}
for syn in
    wn.all_synsets('n'): for
    car_syn in car_synsets:
        sim =
        syn.path_similarity(car_syn) if
        sim is not None:
            similarities[syn.name()] = sim
close_words = sorted(similarities.items(), key=lambda x: x[1],
reverse=True) [:10]
print("\n17. Top Related Words to 'car':",

close_words) # 18. Words that are monosemous

(only one sense)

mono_words = [lemma for lemma in lemmas if
len(wn.synsets(lemma)) == 1] print("\n18. Monosemous Words:",
mono_words[:10])

# 19. Longest synonym chain (most synonyms in synset)
```

```
longest_chain = max(wn.all_synsets(), key=lambda s:
len(s.lemma_names())) print("\n19. Longest Synonym Chain:",
longest_chain.name(), longest_chain.lemma_names())
```

```
# 20. Words with polarity conflict (positive & negative antonyms)

 conflicting_words
  = [] for lemma in
lemmas: has_pos =
  False has_neg =
          False
   for syn in
      wn.synsets(lemma): for l
      in syn.lemmas():
         for ant in l.antonyms():
            if 'positive' in ant.name().lower():
               has_pos = True
            if 'negative' in ant.name().lower():
               has_neg = True
   if has_pos and has_neg:
      conflicting_words.append(lemma)
print("\n20. Words with Polarity Conflict:", conflicting_words[:10])
```

FINAL CODE

```
import pandas as pd
import numpy as np
from nltk.corpus import wordnet as wn

# 1. Top 10 synsets with highest number of
synonyms top_synsets = sorted(wn.all_synsets(),
key=lambda s: len(s.lemma_names()),
reverse=True)[:10]
print("\n1. Top 10 Synsets with Most
Synonyms:") for syn in top_synsets:
    print(syn.name(), syn.lemma_names())

# 2. Word with maximum different senses
lemmas = wn.all_lemma_names()
sense_count = {lemma: len(wn.synsets(lemma)) for lemma in
lemmas} most_senses = max(sense_count, key=sense_count.get)
print("\n2. Word with Most Senses:", most_senses,
sense_count[most_senses])

# 3. All synonyms of a word (e.g.,
happy) word = 'happy'
synonyms = set()
for syn in
    wn.synsets(word): for
    lemma in syn.lemmas():
        synonyms.add(lemma.name())
print("\n3. Synonyms of 'happy':",
synonyms)

# 4. Words having both noun and verb senses
both_nv = [lemma for lemma in lemmas if
wn.synsets(lemma, pos=wn.NOUN) and wn.synsets(lemma,
pos=wn.VERB)] print("\n4. Words with both Noun and Verb
senses:", both_nv[:10])

# 5. Most common hypernyms
hypernym_counter = {}
for syn in wn.all_synsets('n'):
    for hyper in
```

```
    syn.hypernyms():
        hypernym_counter[hyper.name()] =
hypernym_counter.get(hyper.name(), 0) + 1
common_hypernyms = sorted(hypernym_counter.items(),
key=lambda x: x[1], reverse=True)[:10]
print("\n5. Most Common Hypernyms:", common_hypernyms)
```

```
# 6. All pairs of antonyms
antonym_pairs = []
for syn in wn.all_synsets():
    for lemma in
    syn.lemmas():
        if lemma.antonyms():
            antonym_pairs.append((lemma.name(),
            lemma.antonyms()
[0].name()))
print("\n6. Antonym Pairs:", antonym_pairs[:10])

# 7. Words with most antonyms
antonym_count = {}
for lemma in lemmas:
    antonyms = []
    for syn in
        wn.synsets(lemma): for l in
        syn.lemmas():
            antonyms += l.antonyms()
    antonym_count[lemma] =
    len(antonyms)
most_antonyms = max(antonym_count, key=antonym_count.get)
print("\n7. Word with Most Antonyms:", most_antonyms)

# 8. Synset counts per part of speech
pos_counts = {pos: len(list(wn.all_synsets(pos))) for pos in
[wn.NOUN, wn.VERB, wn.ADJ, wn.ADV]}
print("\n8. Synsets per Part of Speech:", pos_counts)

# 9. Synsets with 5+ hyponyms
large_synsets = []
for syn in wn.all_synsets():
    if len(syn.hyponyms()) >= 5:
        large_synsets.append(syn.name())
print("\n9. Synsets with 5+ Hyponyms:", large_synsets[:10])

# 10. All hyponyms of a word (e.g., vehicle)
vehicle_synsets = wn.synsets('vehicle',
pos=wn.NOUN) vehicle_hyponyms = set()
for syn in vehicle_synsets:
```

```
    for hypo in syn.hyponyms():
        vehicle_hyponyms.add(hypo.name())
print("\n10. Hyponyms of 'vehicle':",

vehicle_hyponyms) # 11. Average number of

synonyms per synset
```

```
syn_counts = [len(syn.lemma_names()) for syn in wn.all_synsets()]
avg_synonyms = np.mean(syn_counts)
print("\n11. Average Synonyms per Synset:", avg_synonyms)

# 12. Words that are adjectives and adverbs
adj_adv = [lemma for lemma in lemmas if
wn.synsets(lemma, pos=wn.ADJ) and
wn.synsets(lemma, pos=wn.ADV)] print("\n12. Words
both Adjective and Adverb:", adj_adv[:10])

# 13. Maximum depth in noun hierarchy
max_depth = max(syn.min_depth() for syn in wn.all_synsets('n'))
print("\n13. Maximum Depth in Noun Hierarchy:", max_depth)

# 14. Direct hypernyms of a word (e.g., dog)
dog_synsets = wn.synsets('dog',
pos=wn.NOUN) hypernyms_dog = []
for syn in dog_synsets:
    hypernyms_dog +=
    syn.hypernyms()
print("\n14. Direct Hypernyms of 'dog':", [h.name() for h in
hypernyms_dog])

# 15. Words with multiple hypernyms
multi_hypernym_words = []
for syn in
    wn.all_synsets('n'): if
    len(syn.hypernyms()) >
    1:
        multi_hypernym_words.append(syn.name())
print("\n15. Words with Multiple Hypernyms:",
multi_hypernym_words[:10])

# 16. Synsets with no hypernyms (root concepts)
root_synsets = [syn.name() for syn in wn.all_synsets('n')
if not syn.hypernyms()]
print("\n16. Root Synsets:", root_synsets[:10])

# 17. Top 10 words closely related to 'car' (path similarity)
car_synsets = wn.synsets('car')
similarities = {}
```

```
for syn in
    wn.all_synsets('n'): for
    car_syn in car_synsets:
        sim =
        syn.path_similarity(car_syn) if
        sim is not None:
            similarities[syn.name()] = sim
```

```
close_words = sorted(similarities.items(), key=lambda x: x[1],
reverse=True)[:10]
print("\n17. Top Related Words to 'car':", close_words)

# 18. Words that are monosemous (only one sense)
mono_words = [lemma for lemma in lemmas if
len(wn.synsets(lemma))
== 1]
print("\n18. Monosemous Words:", mono_words[:10])

# 19. Longest synonym chain (most synonyms in synset)
longest_chain = max(wn.all_synsets(), key=lambda s:
len(s.lemma_names()))
print("\n19. Longest Synonym Chain:", longest_chain.name(),
longest_chain.lemma_names())

# 20. Words with polarity conflict (positive & negative antonyms)
conflicting_words = []
for lemma in lemmas:
    has_pos = False
    has_neg = False
    for syn in
        wn.synsets(lemma): for l in
        syn.lemmas():
            for ant in l.antonyms():
                if 'positive' in ant.name().lower():
                    has_pos = True
                if 'negative' in ant.name().lower():
                    has_neg = True
    if has_pos and has_neg:
        conflicting_words.append(lemma)
print("\n20. Words with Polarity Conflict:", conflicting_words[:10])
```