1a

```
import re

text = """
https://twitter.com/elon_musk, https://twitter.com/user_123, and
https://twitter.com/user_456
"""

pattern = r"https://twitter\.com/([A-Za-z0-9_]+)"

twitter_handles = re.findall(pattern, text)

print("Extracted Twitter Handles:",twitter_handles)
```

```
Extracted Twitter Handles: ['elon_musk', 'user_123', 'user_456']
```

1b

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: click in
/usr/local/lib/python3.11/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in
/usr/local/lib/python3.11/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
```

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')

text = """Natural language processing is a field of artificial
intelligence that focuses on the interaction between computers and
humans."""


tokens = word_tokenize(text)
stop_words = set(stopwords.words("english"))
filtered_tokens = [word for word in tokens if word.lower() not in
stop_words]
```

```python
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]

print("Original Text:",text)
print("\nTokens:",tokens)
print("\nFiltered Tokens (Without Stop Words):",filtered_tokens)
print("\nStemmed Tokens:",stemmed_tokens)
```

```
Original Text: Natural language processing is a field of artificial
intelligence that focuses on the interaction between computers and
humans.

Tokens: ['Natural', 'language', 'processing', 'is', 'a', 'field',
'of', 'artificial', 'intelligence', 'that', 'focuses', 'on', 'the',
'interaction', 'between', 'computers', 'and', 'humans', '.']

Filtered Tokens (Without Stop Words): ['Natural', 'language',
'processing', 'field', 'artificial', 'intelligence', 'focuses',
'interaction', 'computers', 'humans', '.']

Stemmed Tokens: ['natur', 'languag', 'process', 'field', 'artifici',
'intellig', 'focus', 'interact', 'comput', 'human', '.']

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

2a

```python
import nltk
import matplotlib.pyplot as plt
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from collections import Counter

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')

text = """
Natural language processing is a branch of artificial intelligence
that helps computers
understand, interpret, and manipulate human language. NLP is used in
applications such as
speech recognition, translation, and sentiment analysis.
```

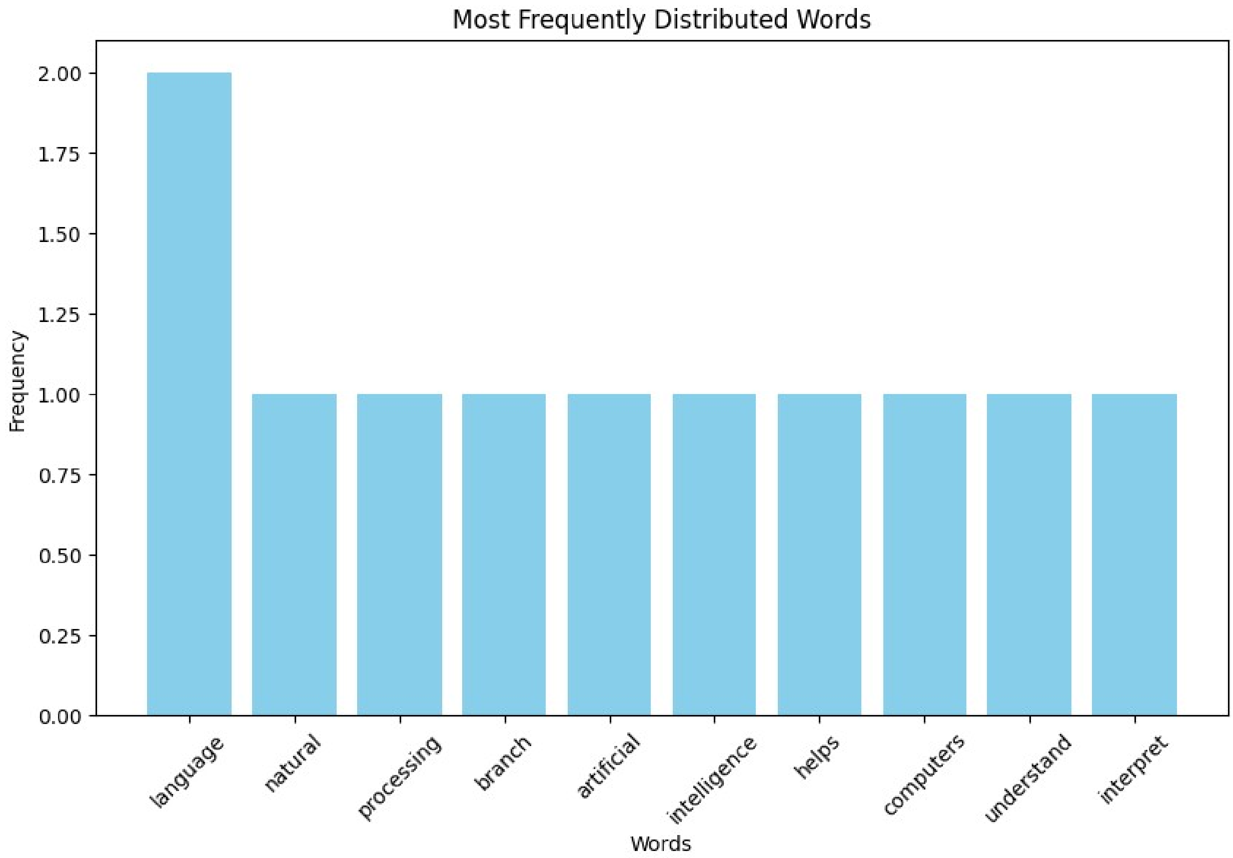
```
"""

tokens = word_tokenize(text)
stop_words = set(stopwords.words('english'))
filtered_tokens = [word.lower() for word in tokens if word.isalpha()
and word.lower() not in stop_words]

word_count = Counter(filtered_tokens)
most_common_words = word_count.most_common(10)
words, counts = zip(*most_common_words)

plt.figure(figsize=(10, 6))
plt.bar(words, counts, color='skyblue')
plt.title("Most Frequently Distributed Words")
plt.xlabel("Words")
plt.ylabel("Frequency")
plt.xticks(rotation=45)
plt.show()
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```



2b

```
import nltk
import re
from nltk.tokenize import word_tokenize

nltk.download('punkt')

text = """
Natural language processing is an interdisciplinary field that uses
computational techniques
to analyze and understand human language.
"""

nltk_tokens = word_tokenize(text)
split_tokens = text.split()
regex_tokens = re.findall(r'\b\w+\b', text)

print("NLTK word_tokenize:", nltk_tokens)
print("Python split():", split_tokens)
print("Regex-based split:", regex_tokens)

print("\nAnalysis of Differences:")
print(f"1. NLTK word_tokenize: {len(nltk_tokens)} tokens, includes
punctuations.")
print(f"2. Python split(): {len(split_tokens)} tokens, splits by
spaces only.")
print(f"3. Regex-based split: {len(regex_tokens)} tokens, removes
punctuations while splitting.")
```

```
NLTK word_tokenize: ['Natural', 'language', 'processing', 'is', 'an',
'interdisciplinary', 'field', 'that', 'uses', 'computational',
'techniques', 'to', 'analyze', 'and', 'understand', 'human',
'language', '.']
Python split(): ['Natural', 'language', 'processing', 'is', 'an',
'interdisciplinary', 'field', 'that', 'uses', 'computational',
'techniques', 'to', 'analyze', 'and', 'understand', 'human',
'language.']
Regex-based split: ['Natural', 'language', 'processing', 'is', 'an',
'interdisciplinary', 'field', 'that', 'uses', 'computational',
'techniques', 'to', 'analyze', 'and', 'understand', 'human',
'language']

Analysis of Differences:
1. NLTK word_tokenize: 18 tokens, includes punctuations.
2. Python split(): 17 tokens, splits by spaces only.
3. Regex-based split: 17 tokens, removes punctuations while splitting.

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

3a

```
!pip install svgling
```

```
Collecting svgling
  Downloading svgling-0.5.0-py3-none-any.whl.metadata (7.4 kB)
Collecting svgwrite (from svgling)
  Downloading svgwrite-1.4.3-py3-none-any.whl.metadata (8.8 kB)
Downloading svgling-0.5.0-py3-none-any.whl (31 kB)
Downloading svgwrite-1.4.3-py3-none-any.whl (67 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/67.1 kB ? eta -:--:--
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.1/67.1 kB 3.3 MB/s eta
0:00:00
```

```
import nltk
import svgling
from nltk import pos_tag, word_tokenize
from nltk.tree import Tree

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

text = "Elon Musk founded SpaceX in 2002 to revolutionize space
```

```
technology."

tokens = word_tokenize(text)

pos_tags = pos_tag(tokens)
print("Part-of-Speech Tags:", pos_tags)

grammar = "NP: {<DT>?<JJ>*<NN>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(pos_tags)

print("\nParsed Tree:")
svgling.draw_tree(result)
```

```
Part-of-Speech Tags: [('Elon', 'NNP'), ('Musk', 'NNP'), ('founded',
'VBD'), ('SpaceX', 'NNP'), ('in', 'IN'), ('2002', 'CD'), ('to', 'TO'),
('revolutionize', 'VB'), ('space', 'NN'), ('technology', 'NN'), ('.',
'.')]

Parsed Tree:

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

3b

```
import re

# Input text
text = ''' Born Elon Reeve Musk
June 28, 1971 (age 50)
Pretoria, Transvaal, South Africa Citizenship
South Africa
Education University of Pennsylvania (BS, BA)
Title Founder, CEO and Chief Engineer of SpaceX
CEO and product architect of Tesla, Inc.
```

```
Founder of The Boring Company and X.com (now part of PayPal)
Co-founder of Neuralink, OpenAI, and Zip2
Spouse(s) Justine Wilson (m. 2000; div. 2008) '''

names = re.search(r"Born\s(.+)\n", text)
name = names.group(1) if names else "Not found"

ages = re.search(r"\(age\s(\d+)\)", text)
age = ages.group(1) if ages else "Not found"

dobs = re.search(r"Born.*\n(.+)\(", text)
dob = dobs.group(1).strip() if dobs else "Not found"

pobs = re.search(r"\)\n(.+),\sSouth Africa", text)
pob = pobs.group(1).strip() if pobs else "Not found"

educations = re.search(r"Education\s(.+)\n", text)
education = educations.group(1) if educations else "Not found"

titles = re.findall(r"Title\s(.+)\n", text)
title = titles if titles else ["Not found"]

print(f"Name: {name}")
print(f"Age: {age}")
print(f"Date of Birth: {dob}")
print(f"Place of Birth: {pob}")
print(f"Education: {education}")
print(f"Titles:")
for title in titles:
    print(title)
```

```
Name: Elon Reeve Musk
Age: 50
Date of Birth: June 28, 1971
Place of Birth: Pretoria, Transvaal
Education: University of Pennsylvania (BS, BA)
Titles:
Founder, CEO and Chief Engineer of SpaceX
```

4a

```
import nltk
import string
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.util import ngrams

nltk.download('punkt')
nltk.download('stopwords')

text = "Artificial intelligence has made significant advancements, but
it still struggles to understand human emotions and context, limiting
its ability to interact naturally."

tokens = word_tokenize(text)
stop_words = set(stopwords.words("english"))
filtered_tokens = [word.lower() for word in tokens if word.isalnum()
and word.lower() not in stop_words]

bigrams = list(ngrams(filtered_tokens, 2))
trigrams = list(ngrams(filtered_tokens, 3))

print("Bigrams:", bigrams)
print("Trigrams:", trigrams)
```

```
Bigrams: [('artificial', 'intelligence'), ('intelligence', 'made'),
('made', 'significant'), ('significant', 'advancements'),
('advancements', 'still'), ('still', 'struggles'), ('struggles',
'understand'), ('understand', 'human'), ('human', 'emotions'),
('emotions', 'context'), ('context', 'limiting'), ('limiting',
'ability'), ('ability', 'interact'), ('interact', 'naturally')]
Trigrams: [('artificial', 'intelligence', 'made'), ('intelligence',
'made', 'significant'), ('made', 'significant', 'advancements'),
('significant', 'advancements', 'still'), ('advancements', 'still',
'struggles'), ('still', 'struggles', 'understand'), ('struggles',
'understand', 'human'), ('understand', 'human', 'emotions'), ('human',
'emotions', 'context'), ('emotions', 'context', 'limiting'),
('context', 'limiting', 'ability'), ('limiting', 'ability',
'interact'), ('ability', 'interact', 'naturally')]
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

4b

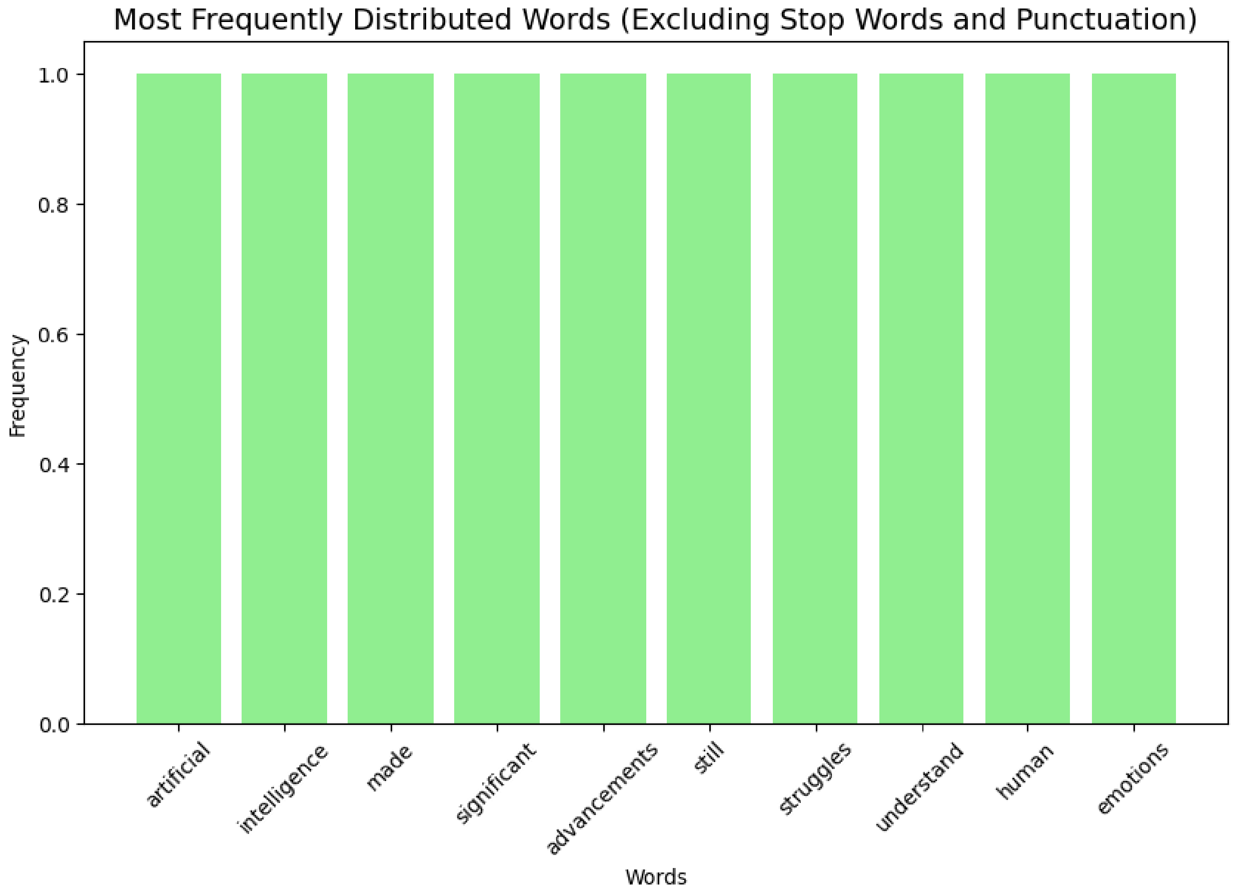
```
from collections import Counter
import matplotlib.pyplot as plt

text = """Artificial intelligence has made significant advancements,
but it still struggles to understand human emotions and context,
limiting its ability to interact naturally."""

tokens = word_tokenize(text)
stop_words = set(stopwords.words('english'))
filtered_tokens = [word.lower() for word in tokens if word.isalpha()
and word.lower() not in stop_words]
```

```python
word_counts = Counter(filtered_tokens)
most_common_words = word_counts.most_common(10)
words, counts = zip(*most_common_words)

plt.figure(figsize=(10, 6))
plt.bar(words, counts, color='lightgreen')
plt.title("Most Frequently Distributed Words (Excluding Stop Words and
Punctuation)", fontsize=14)
plt.xlabel("Words")
plt.ylabel("Frequency")
plt.xticks(rotation=45)
plt.show()
```



5a

```python
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
```

```python
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')

text = "The quick brown fox jumps over the lazy dog"

tokens = word_tokenize(text)
pos_tags = nltk.pos_tag(tokens)
lemmatizer = WordNetLemmatizer()

lemmatized_words = []
for word, tag in pos_tags:
    if tag.startswith('J'):
        pos = wordnet.ADJ
    elif tag.startswith('V'):
        pos = wordnet.VERB
    elif tag.startswith('N'):
        pos = wordnet.NOUN
    elif tag.startswith('R'):
        pos = wordnet.ADV
    else:
        pos = wordnet.NOUN

    lemmatized_words.append(lemmatizer.lemmatize(word, pos))

print("Original Tokens:", tokens)
print("POS Tags:", pos_tags)
print("Lemmatized Tokens:", lemmatized_words)
```

```
Original Tokens: ['The', 'quick', 'brown', 'fox', 'jumps', 'over',
'the', 'lazy', 'dog']
POS Tags: [('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox',
'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('lazy',
'JJ'), ('dog', 'NN')]
Lemmatized Tokens: ['The', 'quick', 'brown', 'fox', 'jump', 'over',
'the', 'lazy', 'dog']
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

5b

```python
import nltk
from nltk.tokenize import word_tokenize

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

text = "The quick brown fox jumps over the lazy dog."

tokens = word_tokenize(text)
pos_tags = nltk.pos_tag(tokens)

# Display POS tags
print("POS Tags:", pos_tags)
```

```
POS Tags: [('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox',
'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('lazy',
'JJ'), ('dog', 'NN'), ('.', '.')]
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

6a

```python
import nltk
from nltk.stem import PorterStemmer, LancasterStemmer, SnowballStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

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

text = "The quick brown foxes were jumping over the lazy dogs in the 
garden."

tokens = word_tokenize(text)

porter = PorterStemmer()
lancaster = LancasterStemmer()
snowball = SnowballStemmer("english")
lemmatizer = WordNetLemmatizer()

print("Original Tokens:", tokens)
print("\nStemmed Words:")
print("Porter Stemmer:", [porter.stem(word) for word in tokens])
print("Lancaster Stemmer:", [lancaster.stem(word) for word in tokens])
print("Snowball Stemmer:", [snowball.stem(word) for word in tokens])
print("\nLemmatized Words:")
print("WordNet Lemmatizer:", [lemmatizer.lemmatize(word) for word in tokens])
print("WordNet Lemmatizer (Verb):", [lemmatizer.lemmatize(word, pos='v') for word in tokens])
```

```
Original Tokens: ['The', 'quick', 'brown', 'foxes', 'were', 'jumping', 'over', 'the', 'lazy', 'dogs', 'in', 'the', 'garden', '.']

Stemmed Words:
Porter Stemmer: ['the', 'quick', 'brown', 'fox', 'were', 'jump', 'over', 'the', 'lazi', 'dog', 'in', 'the', 'garden', '.']
Lancaster Stemmer: ['the', 'quick', 'brown', 'fox', 'wer', 'jump', 'ov', 'the', 'lazy', 'dog', 'in', 'the', 'gard', '.']
Snowball Stemmer: ['the', 'quick', 'brown', 'fox', 'were', 'jump', 'over', 'the', 'lazi', 'dog', 'in', 'the', 'garden', '.']

Lemmatized Words:
WordNet Lemmatizer: ['The', 'quick', 'brown', 'fox', 'were', 'jumping', 'over', 'the', 'lazy', 'dog', 'in', 'the', 'garden', '.']
WordNet Lemmatizer (Verb): ['The', 'quick', 'brown', 'fox', 'be', 'jump', 'over', 'the', 'lazy', 'dog', 'in', 'the', 'garden', '.']

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

6b

```python
import spacy

nlp = spacy.load("en_core_web_sm")

text = "The quick brown fox jumps over the lazy dog."

doc = nlp(text)

print(f"{'Token':<10} {'Dependency':<20} {'Head':<10} {'Children'}\n\n")
for token in doc:
    children = [child.text for child in token.children]
    print(f"{token.text:<10} {token.dep_:<20} {token.head.text:<10} {children}")
```

```
spacy.displacy.render(doc, style="dep", jupyter=True)

Token      Dependency          Head      Children


The        det                 fox       []
quick      amod                fox       []
brown      amod                fox       []
fox        nsubj               jumps     ['The', 'quick', 'brown']
jumps      ROOT                jumps     ['fox', 'over', '.']
over       prep                jumps     ['dog']
the        det                 dog       []
lazy       amod                dog       []
dog        pobj                over      ['the', 'lazy']
.          punct               jumps     []

<IPython.core.display.HTML object>
```

7

```
import nltk
from nltk.util import ngrams
from nltk.tokenize import word_tokenize
from collections import Counter
import string

nltk.download('punkt')
nltk.download('punkt_tab')

documents = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one."
]

all_tokens = []
for doc in documents:
    tokens = word_tokenize(doc.lower())
    tokens = [token for token in tokens if token.isalnum()]
    all_tokens.extend(tokens)

bigrams = list(ngrams(all_tokens, 2))
bigram_freq = Counter(bigrams)
total_unique_bigrams = len(bigram_freq)
top_5_bigrams = bigram_freq.most_common(5)

print("Total unique bi-grams:",total_unique_bigrams)
print("Top 5 most common bi-grams:",top_5_bigrams)
```

```
Total unique bi-grams: 13
Top 5 most common bi-grams: [(('is', 'the'), 3), (('this', 'is'), 2),
(('the', 'first'), 1), (('first', 'document'), 1), (('document',
'this'), 1)]

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

8

```python
import nltk
from nltk import bigrams
from nltk.tokenize import word_tokenize
from collections import Counter

nltk.download('punkt')

text = """
Artificial intelligence is transforming technology.
Machine learning and deep learning are subsets of AI.
AI applications in healthcare and education are remarkable.
"""

tokens = word_tokenize(text)
unigram_count = Counter(tokens)
bigram_count = Counter(bigrams(tokens))
vocab_size = len(unigram_count)

print("Bigram probabilities with Laplace smoothing:")
for (w1, w2), count in bigram_count.items():
    prob = (bigram_count[(w1, w2)] + 1) / (unigram_count[w1] +
vocab_size)
    print(f"P({w2} | {w1}) = {prob:.4f}")

unseen_w1, unseen_w2 = "artificial", "intelligence"
unseen_prob = (bigram_count[(unseen_w1, unseen_w2)] + 1) /
(unigram_count[unseen_w1] + vocab_size)
print(f"\nProbability of unseen bigram ('{unseen_w1}', '{unseen_w2}'):
{unseen_prob:.4f}")
```

```
Bigram probabilities with Laplace smoothing:
P(intelligence | Artificial) = 0.1000
P(is | intelligence) = 0.1000
P(transforming | is) = 0.1000
P(technology | transforming) = 0.1000
P(. | technology) = 0.1000
P(Machine | .) = 0.0909
P(learning | Machine) = 0.1000
```

```
P(and | learning) = 0.0952
P(deep | and) = 0.0952
P(learning | deep) = 0.1000
P(are | learning) = 0.0952
P(subsets | are) = 0.0952
P(of | subsets) = 0.1000
P(AI | of) = 0.1000
P(. | AI) = 0.0952
P(AI | .) = 0.0909
P(applications | AI) = 0.0952
P(in | applications) = 0.1000
P(healthcare | in) = 0.1000
P(and | healthcare) = 0.1000
P(education | and) = 0.0952
P(are | education) = 0.1000
P(remarkable | are) = 0.0952
P(. | remarkable) = 0.1000

Probability of unseen bigram ('artificial', 'intelligence'): 0.0526

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

10

```
def min_edit_distance(source, target):
    m, n = len(source), len(target)
    dp = [[0] * (n + 1) for _ in range(m + 1)]

    for i in range(m + 1):
        dp[i][0] = i
    for j in range(n + 1):
        dp[0][j] = j

    for i in range(1, m + 1):
        for j in range(1, n + 1):
            cost = 0 if source[i - 1] == target[j - 1] else 1
            dp[i][j] = min(dp[i - 1][j] + 1,
                           dp[i][j - 1] + 1,
                           dp[i - 1][j - 1] + cost)
    return dp[m][n]


source1, target1 = "intention", "execution"
print(f"Edit distance between '{source1}' and '{target1}':
{min_edit_distance(source1, target1)}")
```

```
Edit distance between 'intention' and 'execution': 5
```

11

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer

nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger')
nltk.download('vader_lexicon')


text = "The beautiful garden has lovely flowers."

tokens = nltk.word_tokenize(text)
pos_tags = nltk.pos_tag(tokens)

nouns = [word for word, tag in pos_tags if tag in ('NN', 'NNS', 'NNP',
'NNPS')]
adjectives = [word for word, tag in pos_tags if tag in ('JJ', 'JJR',
'JJS')]

sia = SentimentIntensityAnalyzer()
sentiment = sum(sia.polarity_scores(adj)['compound'] for adj in
adjectives)

print("Nouns:", nouns)
print("Adjectives:", adjectives)
print("Sentiment Score of Adjectives:",round(sentiment,2))
```

```
Nouns: ['garden', 'flowers']
Adjectives: ['beautiful', 'lovely']
Sentiment Score of Adjectives: 1.19
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

12

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

corpus = [
    "The Cat in the Hat",
    "The quick brown fox jumps over the lazy dog",
    "The cat jumped over the fence",
    "A brown dog jumped over the cat"
]

vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(corpus)
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(),
columns=vectorizer.get_feature_names_out())

print("TF-IDF Matrix:\n",tfidf_df)
```

```
TF-IDF Matrix:
       brown       cat       dog     fence       fox       hat
in  \
0  0.000000  0.341340  0.000000  0.000000  0.000000  0.534776
0.534776
1  0.303688  0.000000  0.303688  0.000000  0.385189  0.000000
0.000000
2  0.000000  0.339934  0.000000  0.532572  0.000000  0.000000
0.000000
3  0.458882  0.371504  0.458882  0.000000  0.000000  0.000000
0.000000

     jumped     jumps      lazy      over     quick       the
0  0.000000  0.000000  0.000000  0.000000  0.000000  0.558136
1  0.000000  0.385189  0.385189  0.245861  0.385189  0.402016
2  0.419886  0.000000  0.000000  0.339934  0.000000  0.555836
3  0.458882  0.000000  0.000000  0.371504  0.000000  0.303729
```