```
In [1]: # Assignment - A7  |  Name : Pratik Pingale  |  Roll No : 19CO056
```

Sample Sentences

```
In [2]: sentence1 = "I will walk 500 miles and I would walk 500 more. Just to be the man who w
                    "a thousand miles to fall down at your door!"
        sentence2 = "I played the play playfully as the players were playing in the play with
```

Tokenization

```
In [3]: from nltk import word_tokenize, sent_tokenize

        print('Tokenized words:', word_tokenize(sentence1))
        print('\nTokenized sentences:', sent_tokenize(sentence1))
```

```
Tokenized words: ['I', 'will', 'walk', '500', 'miles', 'and', 'I', 'would', 'walk',
'500', 'more', '.', 'Just', 'to', 'be', 'the', 'man', 'who', 'walks', 'a', 'thousan
d', 'miles', 'to', 'fall', 'down', 'at', 'your', 'door', '!']

Tokenized sentences: ['I will walk 500 miles and I would walk 500 more.', 'Just to be
the man who walks a thousand miles to fall down at your door!']
```

POS Tagging

```
In [4]: from nltk import pos_tag

        token = word_tokenize(sentence1) + word_tokenize(sentence2)
        tagged = pos_tag(token)

        print("Tagging Parts of Speech:", tagged)
```

```
Tagging Parts of Speech: [('I', 'PRP'), ('will', 'MD'), ('walk', 'VB'), ('500', 'C
D'), ('miles', 'NNS'), ('and', 'CC'), ('I', 'PRP'), ('would', 'MD'), ('walk', 'VB'),
('500', 'CD'), ('more', 'JJR'), ('.', '.'), ('Just', 'NNP'), ('to', 'TO'), ('be', 'V
B'), ('the', 'DT'), ('man', 'NN'), ('who', 'WP'), ('walks', 'VBZ'), ('a', 'DT'), ('th
ousand', 'NN'), ('miles', 'NNS'), ('to', 'TO'), ('fall', 'VB'), ('down', 'RP'), ('a
t', 'IN'), ('your', 'PRP$'), ('door', 'NN'), ('!', '.'), ('I', 'PRP'), ('played', 'VB
D'), ('the', 'DT'), ('play', 'NN'), ('playfully', 'RB'), ('as', 'IN'), ('the', 'DT'),
('players', 'NNS'), ('were', 'VBD'), ('playing', 'VBG'), ('in', 'IN'), ('the', 'DT'),
('play', 'NN'), ('with', 'IN'), ('playfullness', 'NN')]
```

Stop-Words Removal

```
In [5]: from nltk.corpus import stopwords

        stop_words = stopwords.words('english')

        token = word_tokenize(sentence1)
        cleaned_token = []

        for word in token:
            if word not in stop_words:
                cleaned_token.append(word)
```

```
print('Unclean version:', token)
print('\nCleaned version:', cleaned_token)
```

```
Unclean version: ['I', 'will', 'walk', '500', 'miles', 'and', 'I', 'would', 'walk',
'500', 'more', '.', 'Just', 'to', 'be', 'the', 'man', 'who', 'walks', 'a', 'thousan
d', 'miles', 'to', 'fall', 'down', 'at', 'your', 'door', '!']

Cleaned version: ['I', 'walk', '500', 'miles', 'I', 'would', 'walk', '500', '.', 'Jus
t', 'man', 'walks', 'thousand', 'miles', 'fall', 'door', '!']
```

Stemming

In [6]:
```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

token = word_tokenize(sentence2)

stemmed = [stemmer.stem(word) for word in token]
print(" ".join(stemmed))
```

```
i play the play play as the player were play in the play with playful
```

Lemmatization

In [7]:
```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

token = word_tokenize(sentence2)

lemmatized_output = [lemmatizer.lemmatize(word) for word in token]
print(" ".join(lemmatized_output))
```

```
I played the play playfully a the player were playing in the play with playfullness
```

In [1]:
```
import math
from collections import Counter
```

In [2]:
```
# Sample corpus of documents
corpus = [
'The quick brown fox jumps over the lazy dog',
'The brown fox is quick',
'The lazy dog is sleeping'
]
```

In [3]:
```
# Tokenize the documents
tokenized_docs = [doc.lower().split() for doc in corpus]
```

In [4]:
```
# Count the term frequency for each document
tf_docs = [Counter(tokens) for tokens in tokenized_docs]
```

In [6]:
```
n_docs = len(corpus)
idf = {}
for tokens in tokenized_docs:
    for token in set(tokens):
        idf[token] = idf.get(token, 0) + 1
for token in idf:
    idf[token] = math.log(n_docs / idf[token])
```

In [7]:
```
tfidf_docs = []
for tf_doc in tf_docs:
    tfidf_doc = {}
    for token, freq in tf_doc.items():
        tfidf_doc[token] = freq * idf[token]
    tfidf_docs.append(tfidf_doc)
```

In [9]:
```
# Print the resulting TF-IDF representation for each document
for i, tfidf_doc in enumerate(tfidf_docs):
    print(f"Document {i+1}: {tfidf_doc}")
```

```
Document 1: {'the': 0.0, 'quick': 0.4054651081081644, 'brown': 0.4054651081081644, 'f
ox': 0.4054651081081644, 'jumps': 1.0986122886681098, 'over': 1.0986122886681098, 'la
zy': 0.4054651081081644, 'dog': 0.4054651081081644}
Document 2: {'the': 0.0, 'brown': 0.4054651081081644, 'fox': 0.4054651081081644, 'i
s': 0.4054651081081644, 'quick': 0.4054651081081644}
Document 3: {'the': 0.0, 'lazy': 0.4054651081081644, 'dog': 0.4054651081081644, 'is':
0.4054651081081644, 'sleeping': 1.0986122886681098}
```

In [ ]: