7) Text Analytics

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of documents by calculating Term Frequency and Inverse DocumentFrequency.

In [3]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk import pos_tag
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
# Download all required NLTK data
def download_nltk_resources():
    resources = {
        'punkt': 'tokenizers/punkt',
        'stopwords': 'corpora/stopwords',
        'averaged_perceptron_tagger': 'taggers/averaged_perceptron_tagger',
        'wordnet': 'corpora/wordnet',
        'omw-1.4': 'corpora/omw-1.4'
    }

    for resource, path in resources.items():
        try:
            nltk.data.find(path)
        except LookupError:
            print(f"Downloading {resource}...")
            nltk.download(resource)

download_nltk_resources()

# Sample document
document = """The quick brown fox jumps over the lazy dog. Dogs are great pets, but foxes are wild animals. Foxes and dogs have different behaviors."""

# 1. Tokenization
try:
    tokens = word_tokenize(document)
    print("Tokenization:")
    print(tokens)
    print("\n" + "="*50 + "\n")
except Exception as e:
    print(f"Tokenization failed: {e}")
    tokens = document.split()  # fallback to simple whitespace tokenization

# 2. POS Tagging
pos_tags = pos_tag(tokens)
print("POS Tagging:")
print(pos_tags)
print("\n" + "="*50 + "\n")

# 3. Stop Words Removal
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words and word.isalpha()]
print("After Stop Words Removal:")
print(filtered_tokens)
print("\n" + "="*50 + "\n")

# 4. Stemming
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]
print("After Stemming:")
print(stemmed_tokens)
print("\n" + "="*50 + "\n")

# 5. Lemmatization
lemmatizer = WordNetLemmatizer()

def get_wordnet_pos(treebank_tag):
    tag = treebank_tag[0].upper() if treebank_tag else 'N'
    tag_dict = {"J": 'a', "N": 'n', "V": 'v', "R": 'r'}
    return tag_dict.get(tag, 'n')

pos_tags_filtered = pos_tag(filtered_tokens)
lemmatized_tokens = []
for word, tag in pos_tags_filtered:
    lemmatized_tokens.append(lemmatizer.lemmatize(word, pos=get_wordnet_pos(tag)))

print("After Lemmatization:")
print(lemmatized_tokens)
print("\n" + "="*50 + "\n")

# 6. TF-IDF Representation
corpus = [
    " ".join(lemmatized_tokens),
    "dog make wonderful companion",
    "fox live forest"
]

try:
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(corpus)
    df_tfidf = pd.DataFrame(tfidf_matrix.toarray(),
                            columns=vectorizer.get_feature_names_out(),
                            index=["Document 1", "Document 2", "Document 3"])

    print("TF-IDF Representation:")
    print(df_tfidf)
except Exception as e:
    print(f"TF-IDF calculation failed: {e}")
```

```
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\saval\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     C:\Users\saval\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger_eng.zip.
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\saval\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\saval\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\saval\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
Downloading punkt...
Downloading wordnet...
Downloading omw-1.4...
Tokenization:
['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', '.', 'Dogs', 'are', 'great', 'pets', ',', 'but', 'foxes', 'are', 'wild', 'animals', '.', 'Foxes', 'and', 'dogs', 'have', 'different', 'behaviors', '.']

==================================================

POS Tagging:
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('lazy', 'JJ'), ('dog', 'NN'), ('.', '.'), ('Dogs', 'NNS'), ('are', 'VBP'), ('great', 'JJ'), ('pets', 'NNS'), (',', ','),
('but', 'CC'), ('foxes', 'NNS'), ('are', 'VBP'), ('wild', 'JJ'), ('animals', 'NNS'), ('.', '.'), ('Foxes', 'NNS'), ('and', 'CC'), ('dogs', 'NNS'), ('have', 'VBP'), ('different', 'JJ'), ('behaviors', 'NNS'), ('.', '.')]

==================================================

After Stop Words Removal:
['quick', 'brown', 'fox', 'jumps', 'lazy', 'dog', 'Dogs', 'great', 'pets', 'foxes', 'wild', 'animals', 'Foxes', 'dogs', 'different', 'behaviors']

==================================================

After Stemming:
['quick', 'brown', 'fox', 'jump', 'lazi', 'dog', 'dog', 'great', 'pet', 'fox', 'wild', 'anim', 'fox', 'dog', 'differ', 'behavior']

==================================================

After Lemmatization:
['quick', 'brown', 'fox', 'jump', 'lazy', 'dog', 'Dogs', 'great', 'pet', 'fox', 'wild', 'animal', 'Foxes', 'dog', 'different', 'behavior']

==================================================

TF-IDF Representation:
             animal  behavior    brown  companion  different       dog  \
Document 1  0.24524   0.24524  0.24524   0.000000    0.24524  0.373022
Document 2  0.00000   0.00000  0.00000   0.528635    0.00000  0.402040
Document 3  0.00000   0.00000  0.00000   0.000000    0.00000  0.000000

                dogs    forest       fox    foxes    great     jump     lazy  \
Document 1  0.24524  0.000000  0.373022  0.24524  0.24524  0.24524  0.24524
Document 2  0.00000  0.000000  0.000000  0.00000  0.00000  0.00000  0.00000
Document 3  0.00000  0.622766  0.473630  0.00000  0.00000  0.00000  0.00000

                live      make      pet    quick     wild  wonderful
Document 1  0.000000  0.000000  0.24524  0.24524  0.24524   0.000000
```

Document 2	0.000000	0.528635	0.00000	0.00000	0.00000	0.528635
Document 3	0.622766	0.000000	0.00000	0.00000	0.00000	0.000000