Experiment 7

Aim

Plagiarism Detection Using NLP

Theory

Natural Language Processing technologies can be used to effectively to detect plagiarism in texts. NLP distance measures can be applied to detect external plagiarism, i.e., when both the original text as well as the suspicious text are available.

Steps

1. Before using other NLP techniques, we first apply pre-processing techniques to the text. change all the uppercase alphabets to lowercase to generalize tokens across both the texts. Further, Stop-Words like ‘or’, ‘the’ and ‘in’ and punctuations are removed, as these are functional in nature and do not give any extra information about the document.
2. Next, we read the original and the suspicious (possibly plagiarized) documents.
3. The plagiarism content between the two texts is found by calculating the Jaccard similarity coefficient.
4. Another method is finding the Longest Common Subsequence (LCS) in the texts.
5. Evaluate all the scores on the documents in the dataset. There are three types of documents: near copy, lightly revised and heavily revised.

Code

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.corpus import wordnet as wn

doc1 = "When your focus is to improve employee performance, it's essential to encourage ongoing
dialogue between managers and their direct reports. Some companies encourage supervisors to
hold one-on-one meetings with employees as a way to facilitate two-way communication."
doc2 = "When your focus is to improve employee performance, ongoing dialogue between managers
and their direct reports is essential. While performance management often involves conducting
annual performance evaluations, it does involve more than just that."

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words("english"))
word_tokens1 = word_tokenize(doc1)
word_tokens2 = word_tokenize(doc2)

filtered_sentence1 = [w for w in word_tokens1 if not w.lower() in stop_words]
filtered_sentence2 = [w for w in word_tokens2 if not w.lower() in stop_words]
```

```
filtered_sentence1 = []
for w in word_tokens1:
    if w not in stop_words:
        filtered_sentence1.append(w)
filtered_sentence2 = []
for w in word_tokens2:
    if w not in stop_words:
        filtered_sentence2.append(w)

print(word_tokens1)
print(filtered_sentence1)
print(word_tokens2)
print(filtered_sentence2)
s1 = " ".join(filtered_sentence1)
s2 = " ".join(filtered_sentence2)

from nltk.tokenize import RegexpTokenizer

tokenizer = RegexpTokenizer(r"\w+")
s1 = tokenizer.tokenize(s1)
s2 = tokenizer.tokenize(s2)
s1 = " ".join(s1)
s2 = " ".join(s2)
jd_sent_1_2 = nltk.jaccard_distance(set(s1), set(s2))
print(f"Similarity using Jaccard Similarity {(1 - jd_sent_1_2)*100}%")



def lcs(l1, l2):
    s1 = word_tokenize(l1)
    s2 = word_tokenize(l2)
    # storing the dp values
    dp = [[None] * (len(s1) + 1) for i in range(len(s2) + 1)]
    for i in range(len(s2) + 1):
        for j in range(len(s1) + 1):
            if i == 0 or j == 0:
                dp[i][j] = 0
            elif s2[i - 1] == s1[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])
    return dp[len(s2)][len(s1)]



from nltk.tokenize import sent_tokenize, word_tokenize

tokens_o = word_tokenize(doc1)
tokens_p = word_tokenize(doc2)
sent_o = sent_tokenize(doc1)
sent_p = sent_tokenize(doc2)

# maximum length of LCS for a sentence in suspicious text
max_lcs = 0
sum_lcs = 0
for i in sent_p:
    for j in sent_o:
        l = lcs(i, j)
        max_lcs = max(max_lcs, l)
    sum_lcs += max_lcs
    smax_lcs = 0

score = sum_lcs / len(tokens_p)
print(f"Similarity using LCS {score*100}%")
```

Output

```
Aryan D:\..\..\..\NLP ❯❯❯ python 7.py
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Aryan\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
['When', 'your', 'focus', 'is', 'to', 'improve', 'employee', 'performance', ',', 'it', "'s", 'essential', 'to', 'encourage', 'ongoing',
 'dialogue', 'between', 'managers', 'and', 'their', 'direct', 'reports', '.', 'Some', 'companies', 'encourage', 'supervisors', 'to', 'h
old', 'one-on-one', 'meetings', 'with', 'employees', 'as', 'a', 'way', 'to', 'facilitate', 'two-way', 'communication', '.']
['When', 'focus', 'improve', 'employee', 'performance', ',', "'s", 'essential', 'encourage', 'ongoing', 'dialogue', 'managers', 'direct
', 'reports', '.', 'Some', 'companies', 'encourage', 'supervisors', 'hold', 'one-on-one', 'meetings', 'employees', 'way', 'facilitate',
 'two-way', 'communication', '.']
['When', 'your', 'focus', 'is', 'to', 'improve', 'employee', 'performance', ',', 'ongoing', 'dialogue', 'between', 'managers', 'and', '
their', 'direct', 'reports', 'is', 'essential', '.', 'While', 'performance', 'management', 'often', 'involves', 'conducting', 'annual',
 'performance', 'evaluations', ',', 'it', 'does', 'involve', 'more', 'than', 'just', 'that', '.']
['When', 'focus', 'improve', 'employee', 'performance', ',', 'ongoing', 'dialogue', 'managers', 'direct', 'reports', 'essential', '.',
'While', 'performance', 'management', 'often', 'involves', 'conducting', 'annual', 'performance', 'evaluations', ',', 'involve', '.']
Similarity using Jaccard Similarity 91.30434782608697%
Similarity using LCS 94.73684210526315%
```

Conclusion

Hence, plagiarism checker has been performed using Jaccard Similarity and LCS.