60004180017
Deep Gandhi

**Experiment No:7**

# Plagiarism Detection Using NLP

**AIM:** Plagiarism Detection Using NLP

**Theory:**
Natural Language Processing technologies can be used to effectively to detect plagiarism intexts. Here, we see NLP(distance measures) applied to detect external plagiarism, i.e when both the original text as well as the suspicious text are available.
Instructions:

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 punctuationsare removed, as these are functional in nature and do not give any extra information about the document.Import NLTK library
2. Next, we read the original and the suspicious (possibly plagiarised)documents.
3. The plagiarism content between the two texts is found by calculating theJaccard similarity coefficient,
4. Another method is finding the *Longest Common Subsequence* (LCS) in thetexts.
5. Evaluate all the scores on the documents in the dataset. There are three typesof documents: *near copy, lightly revised* and *heavily revised.*

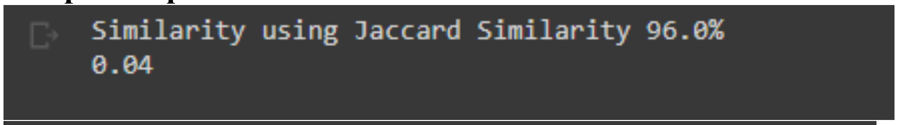
**CODE:**
```
from nltk.corpus import wordnet as
wnimport nltk
from nltk.corpus import stopwords
doc1 = "The legal system is made up of criminal and civil courts and
specialty courts like bankruptcy and family law courts. Each court is
vested with its own jurisdiction. Jurisdiction refers to the types of
cases the court is permitted to rule on. Sometimes, only one type of
court can hear a particular case. For instance, bankruptcy cases can be
ruled
 on only in bankruptcy court. In other situations, it is possible for
more than one court to have jurisdiction. For instance, both a state
and
 federal criminal court could have authority over a criminal case
thatis also considered an offense under federal and state drug laws."
  doc2 = "The legal system is made up of civil courts, criminal courts
and specialty courts, such as family law courts and bankruptcy courts.
Each court has its own jurisdiction, which refers to the cases that the
 court is allowed to hear. In some instances, a case can only be heard
inone type of court. For example, a bankruptcy case must be heard in a
      b ankruptcy court. In other instances, more than one court could
  potentially have jurisdiction. For example, a federal criminal court
  and a state criminal court would each have jurisdiction over a crime
                                                          that is a fe
```

60004180017
Deep Gandhi

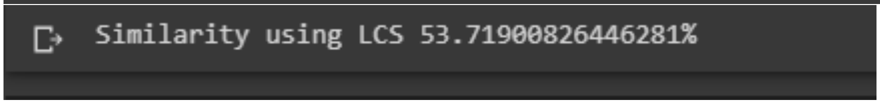
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english'))

word_tokens1 = word_tokenize(doc1)

filtered_sentence = [w for w in word_tokens1 if not w.lower() in
stop_words]

filtered_sentence = []

for w in word_tokens1:
    if w not in stop_words:
        filtered_sentence.append(w)

print(word_tokens1)
print(filtered_sentence)
s1 = " ".join(filtered_sentence)
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_tokenizefrom nltk.tokenize import
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
textmax_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_lc
    smax_lcs=0

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

**Sample Output:**



```
Similarity using Jaccard Similarity 96.0%
0.04
```



```
Similarity using LCS 53.71900826446281%
```

**Conclusion:** Hence, plagiarism checker has been performed using Jaccard Similarityand LCS.