**PRACTICAL NO 1**

**AIM:** Document Indexing and Retrieval

1. Implement an inverted index construction algorithm.

2**.** Build a simple document retrieval system using the constructed index.

**SOLUTION:**

1. Implement an inverted index construction algorithm.

**INPUT:**

document1 = "The quick brown fox jumped over the lazy dog"

document2 = "The lazy dog slept in the sun"

import nltk

from nltk.corpus import stopwords

stop\_words = set(stopwords.words('english'))

# Tokenize and filter stopwords

tokens1 = [word for word in document1.lower().split() if word not in stop\_words]

tokens2 = [word for word in document2.lower().split() if word not in stop\_words]

# Build inverted index and occurrence counts

inverted\_index = {}

occurrences = {"Document 1": {}, "Document 2": {}}

for term in set(tokens1 + tokens2):

inverted\_index[term] = []

if term in tokens1:

inverted\_index[term].append("Document 1")

occurrences["Document 1"][term] = tokens1.count(term)

if term in tokens2:

inverted\_index[term].append("Document 2")

occurrences["Document 2"][term] = tokens2.count(term)

# Print results

print("Inverted Index:", inverted\_index)

print("Occurrences in Document 1:", occurrences["Document 1"])

print("Occurrences in Document 2:", occurrences["Document 2"])

# Print inverted index with occurrences

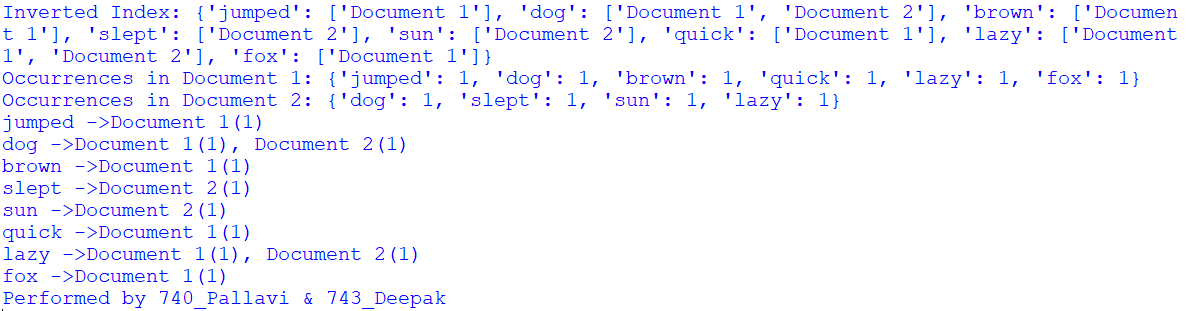
for term, docs in inverted\_index.items():

print(f"{term} ->", end="")

print(", ".join(f"{doc}({occurrences[doc].get(term, 0)})" for doc in docs))

print("Performed by 740\_Pallavi & 743\_Deepak")

**OUTPUT:**



2. Build a simple document retrieval system using the constructed index.

**INPUT:**

import re

from collections import defaultdict

class DocumentRetrievalSystem:

def \_\_init\_\_(self):

self.index = defaultdict(list)

self.documents = []

def add\_documents(self, documents):

for doc\_id, document in enumerate(documents):

self.documents.append(document)

for term in self.tokenize(document):

self.index[term].append(doc\_id)

def search(self, query):

query\_terms = self.tokenize(query)

result\_docs = set(self.index[query\_terms[0]]) if query\_terms and query\_terms[0] in self.index else set()

for term in query\_terms[1:]:

result\_docs &= set(self.index[term])

return [self.documents[doc\_id] for doc\_id in result\_docs]

@staticmethod

def tokenize(text):

return re.findall(r'\b\w+\b', text.lower())

if \_\_name\_\_ == "\_\_main\_\_":

retrieval\_system = DocumentRetrievalSystem()

retrieval\_system.add\_documents([

"This is the first document",

"Python is a popular programming language",

"Document retrieval systems are important"

])

query = "is"

results = retrieval\_system.search(query)

if results:

print(f"Search results for '{query}':")

for result in results:

print("-", result)

else:

print(f"No results found for '{query}'.")

**OUTPUT:**

