In [1]:
```
import json
from nltk.stem import PorterStemmer
import re
```

In [3]:
```
# Load document representatives
with open("./input/documents.json", "r") as f:
    documents = json.load(f)
```

In [4]:
```
ps = PorterStemmer()
```

In [5]:
```
# ---- Build inverted index ----
inverted_index = {}

for doc_name, rep in documents.items():
    for word in rep.keys():
        if word not in inverted_index:
            inverted_index[word] = []
        if doc_name not in inverted_index[word]:
            inverted_index[word].append(doc_name)
```

In [6]:
```
# ---- Query function ----
def search(query, mode="OR"):
    # preprocess query
    words = re.findall(r"\b[a-z]+\b", query.lower())
    stems = [ps.stem(w) for w in words]

    result_sets = []
    for stem in stems:
        if stem in inverted_index:
            result_sets.append(set(inverted_index[stem]))
        else:
            result_sets.append(set())

    if not result_sets:
        return []

    if mode == "AND":
        result = set.intersection(*result_sets)
    else:  # OR search
        result = set.union(*result_sets)

    return list(result)
```

In [8]:
```
print("\nInverted Index:")
for word, docs in inverted_index.items():
    print(f"{word}: {docs}")
```

```
Inverted Index:
machin: ['doc1.txt', 'doc2.txt']
learn: ['doc1.txt', 'doc2.txt']
subset: ['doc1.txt']
artifici: ['doc1.txt', 'doc2.txt', 'doc3.txt']
intellig: ['doc1.txt', 'doc2.txt', 'doc3.txt']
algorithm: ['doc1.txt']
build: ['doc1.txt']
mathemat: ['doc1.txt']
model: ['doc1.txt']
base: ['doc1.txt']
sampl: ['doc1.txt']
data: ['doc1.txt', 'doc3.txt']
known: ['doc1.txt']
train: ['doc1.txt']
make: ['doc1.txt', 'doc3.txt']
predict: ['doc1.txt']
decis: ['doc1.txt', 'doc3.txt']
without: ['doc1.txt']
explicitli: ['doc1.txt']
program: ['doc1.txt']
simul: ['doc2.txt']
human: ['doc2.txt']
process: ['doc2.txt', 'doc3.txt']
ai: ['doc2.txt']
applic: ['doc2.txt']
includ: ['doc2.txt']
natur: ['doc2.txt']
languag: ['doc2.txt']
speech: ['doc2.txt']
recognit: ['doc2.txt']
comput: ['doc2.txt', 'doc4.txt']
vision: ['doc2.txt']
abil: ['doc2.txt']
reason: ['doc2.txt']
self: ['doc2.txt']
correct: ['doc2.txt']
mine: ['doc3.txt']
discov: ['doc3.txt']
pattern: ['doc3.txt']
larg: ['doc3.txt']
dataset: ['doc3.txt']
combin: ['doc3.txt']
statist: ['doc3.txt']
databas: ['doc3.txt']
system: ['doc3.txt']
use: ['doc3.txt']
inform: ['doc3.txt']
help: ['doc3.txt']
network: ['doc4.txt']
connect: ['doc4.txt']
devic: ['doc4.txt']
enabl: ['doc4.txt']
commun: ['doc4.txt']
involv: ['doc4.txt']
protocol: ['doc4.txt']
topolog: ['doc4.txt']
transmiss: ['doc4.txt']
media: ['doc4.txt']
internet: ['doc4.txt']
```

```
largest: ['doc4.txt']
support: ['doc4.txt']
global: ['doc4.txt']
```

In [34]:
```python
# Save inverted index to JSON file
with open("./output/inverted_index.json", "w") as f:
    json.dump(inverted_index, f, indent=4)

print("Inverted index saved to inverted_index.json")
```

```
Inverted index saved to inverted_index.json
```

In [33]:
```python
print("\nSearch Results:")
print("Query: data →", search("data", mode="OR"))
print("Query: computer network protocol →", search("computer network protocol",
print("Query: topological data mining →", search("topological data mining", mode
```

```
Search Results:
Query: data → ['doc1.txt', 'doc3.txt']
Query: computer network protocol → ['doc4.txt']
Query: topological data mining → []
```

In [ ]:
```python

```