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

In [2]:
```
# ---- Load inverted index ----
with open("./input/inverted_index.json", "r") as f:
    inverted_index = json.load(f)
```

In [3]:
```
ps = PorterStemmer()
```

In [4]:
```
# ---- Query function (reusing from Assignment 3) ----
def search(query, mode="OR"):
    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 set()

    if mode == "AND":
        return set.intersection(*result_sets)
    else:
        return set.union(*result_sets)
```

In [5]:
```
# ---- Precision & Recall ----
def precision_recall(query, answer_set, mode="OR"):
    retrieved = search(query, mode)
    relevant = set(answer_set)

    true_positives = retrieved.intersection(relevant)

    precision = len(true_positives) / len(retrieved) if retrieved else 0
    recall = len(true_positives) / len(relevant) if relevant else 0

    return retrieved, precision, recall
```

In [9]:
```
# ---- Example ----
query = "artificial intelligence"
answer_set = ["doc1.txt", "doc2.txt"]  # define manually

retrieved, p, r = precision_recall(query, answer_set, mode="AND")

print(f"Query: {query}")
print(f"Retrieved Docs: {retrieved}")
print(f"Answer Set (Relevant): {answer_set}")
print(f"Precision: {p:.2f}")
print(f"Recall: {r:.2f}")
```

```
Query: artificial intelligence
Retrieved Docs: {'doc3.txt', 'doc1.txt', 'doc2.txt'}
Answer Set (Relevant): ['doc1.txt', 'doc2.txt']
Precision: 0.67
Recall: 1.00
```

In [ ]: