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 [6]:
```
def f_and_e_scores(precision, recall, beta=1.0):
    # F-measure (F1)
    if precision + recall > 0:
        f_measure = 2 * precision * recall / (precision + recall)
    else:
        f_measure = 0

    # E-measure
    if precision == 0 and recall == 0:
        e_measure = 1
    else:
        e_measure = 1 - ((1 + beta**2) * precision * recall) / (beta**2 * precis

    return f_measure, e_measure
```

In [7]:
```
# ---- Example ----
query = "artificial intelligence"
answer_set = ["doc1.txt", "doc2.txt"]  # define manually

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

In [8]:
```
# Case 1: β = 1 (equal weight)
f1, e1 = f_and_e_scores(p, r, beta=1)
```

In [9]:
```
# Case 2: β < 1 (precision weighted, e.g., β=0.5)
f2, e2 = f_and_e_scores(p, r, beta=0.5)
```

In [10]:
```
# Case 3: β > 1 (recall weighted, e.g., β=2)
f3, e3 = f_and_e_scores(p, r, beta=2)
```

In [12]:
```
print(f"\nQuery: {query}")
print(f"Retrieved Docs: {retrieved}")
print(f"Precision: {p:.2f}, Recall: {r:.2f}")
print(f"F-measure (β=1): {f1:.2f}, E-measure: {e1:.2f}")
print(f"F-measure (β=0.5): {f2:.2f}, E-measure: {e2:.2f}")
print(f"F-measure (β=2): {f3:.2f}, E-measure: {e3:.2f}")
```

```
Query: artificial intelligence
Retrieved Docs: {'doc2.txt', 'doc3.txt', 'doc1.txt'}
Precision: 0.67, Recall: 1.00
F-measure (β=1): 0.80, E-measure: 0.20
F-measure (β=0.5): 0.80, E-measure: 0.29
F-measure (β=2): 0.80, E-measure: 0.09
```

In [ ]: