

```

package ir4;

import java.util.*;

public class FEMeasure {

    public static void main(String[] args) {

        Map<String, String> documents = new HashMap<>();

        documents.put("Document1", "Information retrieval (IR) is the process of obtaining information from a collection of resources.");

        documents.put("Document2", "Text analysis involves various techniques for extracting insights from textual data.");

        Map<String, String> queries = new HashMap<>();

        queries.put("Query1", "information retrieval techniques");

        queries.put("Query2", "text analysis process");

        Map<String, List<String>> relevanceJudgments = new HashMap<>();

        relevanceJudgments.put("Query1", Arrays.asList("Document1"));

        relevanceJudgments.put("Query2", Arrays.asList("Document2"));

        for (String queryId : queries.keySet()) {

            String query = queries.get(queryId);

            List<String> relevantDocuments = relevanceJudgments.getOrDefault(queryId, Collections.emptyList());

            Set<String> queryTokens = tokenize(query);

            Set<String> documentTokens = tokenize(documents.get(relevantDocuments.get(0)));

            double precision = calculatePrecision(queryTokens, documentTokens);

            double recall = calculateRecall(queryTokens, documentTokens);

            double fMeasure = calculateFMeasure(precision, recall);

            double eMeasure = calculateEMeasure(precision, recall);

```

```

        System.out.println("Query: " + query);
        System.out.println("Relevant Documents: " + relevantDocuments);
        System.out.println("Precision: " + precision);
        System.out.println("Recall: " + recall);
        System.out.println("F-Measure: " + fMeasure);
        System.out.println("E-Measure: " + eMeasure);
        System.out.println();
    }
}

```

```

private static Set<String> tokenize(String text) {
    return new HashSet<>(Arrays.asList(text.toLowerCase().split("\\s+")));
}

```

```

private static double calculatePrecision(Set<String> queryTokens, Set<String> documentTokens) {
    if (queryTokens.isEmpty()) {
        return 0.0;
    }
    Set<String> intersection = new HashSet<>(queryTokens);
    intersection.retainAll(documentTokens);
    return (double) intersection.size() / queryTokens.size();
}

```

```

private static double calculateRecall(Set<String> queryTokens, Set<String> documentTokens) {
    if (documentTokens.isEmpty()) {
        return 0.0;
    }
    Set<String> intersection = new HashSet<>(queryTokens);
    intersection.retainAll(documentTokens);
    return (double) intersection.size() / documentTokens.size();
}

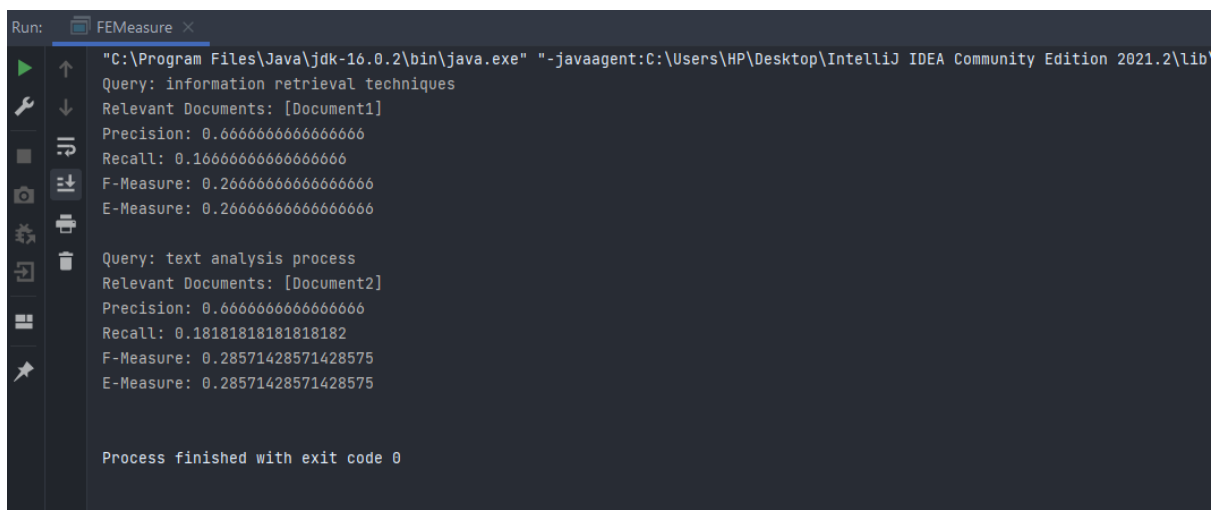
```

```
}
```

```
private static double calculateFMeasure(double precision, double recall) {  
    if (precision + recall == 0) {  
        return 0.0;  
    }  
    return (2 * precision * recall) / (precision + recall);  
}
```

```
private static double calculateEMeasure(double precision, double recall) {  
    if (precision == 0.0 && recall == 0.0) {  
        return 0.0;  
    }  
    return (2 * precision * recall) / (precision + recall);  
}  
}
```

Output:



```
Run: FEMeasure x  
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Users\HP\Desktop\IntelliJ IDEA Community Edition 2021.2\lib  
Query: information retrieval techniques  
Relevant Documents: [Document1]  
Precision: 0.6666666666666666  
Recall: 0.16666666666666666  
F-Measure: 0.26666666666666666  
E-Measure: 0.26666666666666666  
  
Query: text analysis process  
Relevant Documents: [Document2]  
Precision: 0.6666666666666666  
Recall: 0.18181818181818182  
F-Measure: 0.28571428571428575  
E-Measure: 0.28571428571428575  
  
Process finished with exit code 0
```