Report

Team

Name - UID - mail

Mourya Gottipati - U01099603 - gottipati.22@wright.edu
Niranjan Kumar Ramineni - U01065247 - ramineni.15@wright.edu

1. Introduction: The primary goal of this assignment involves the development of a basic Information Retrieval System (IRS) using the Cranfield Dataset. Its objective is to apply concepts such as term generation, indexing and query processing using scikit-learn for vectorization and similarity computation.

2. Data Extraction and Preprocessing

2.1 Data Extraction

- **Cranfield Dataset:** The dataset consists of 1,400 documents and 225 queries which are organized into multiple sections delimited by tags such as (.T) for title, (.A) for author, (.W) for body and (.I) for ID. Only the body (.W) sections are used for both documents and queries in this assignment.
- **Components**:

→ **cran.all**: Contains the entire document collection.
→ **query.txt**: Contains the queries.
→ **qrels.txt**: Contains all relevant documents for each query.

2.2 Preprocessing Steps

- **Tokenization**: Splitting text into tokens while removing irrelevant characters like punctuation.
- **Case-Folding**: Converted all text to lowercase to ensure consistency.
- **Eliminating Stop-Words:** Eliminating common words using "***sklearn.feature_extraction.text.ENGLISH_STOP_WORDS***".

3. Vectorization

3.1 Binary Vectorizer

- **Definition**: It Assigns '1' if a term is present in a document/query and '0' otherwise.
- **Implementation**: " CountVectorizer()" is used from scikit-learn API which enables stop_words and lowercase options in scikit-learn's Binary Vectorizer.

3.2 Custom TF-IDF Vectorizer

- **Term Frequency (TF) Calculation**:
  → Modified TF formula:

$$TF_{modified(t,d)} = \frac{1 + log(count(t,d))}{1 + log(length(d))}$$

  → It uses the impact of document length and reduces the effect of outliers.

- **Inverse Document Frequency (IDF) Calculation**:
  → Modified IDF formula:

$$IDF_{modified(t,d)} = \frac{1}{DF(t)}$$

  → It is the reciprocal of the document frequency.

4. Similarity Calculation

4.1 Cosine Similarity

- **Formula**:

$$\cos(d_i, d_j) = \frac{d_i \bullet q_j}{\| d_i \| \| q_j \|} = \frac{\sum_{k=1}^{n} d_{i_k} q_{j_k}}{\sqrt{\sum_{k=1}^{n} d_{i_k}^2} \sqrt{\sum_{k=1}^{n} q_{j_k}^2}}$$

- It measures the cosine of the angle between two vectors. Values range from 0 (orthogonal) to 1 (identical) and higher values indicate high similarity.

4.2 Euclidean Distance

- **Formula**:

$$Euclidean(d_i, q_j) = \sqrt{\sum_{k=1}^{n} (d_{i,k} - q_{j,k})^2}$$

- It measures the straight-line distance between two vectors. Lower values indicate higher similarity.

5. Retrieval and Evaluation

5.1 Retrieval of Top 10 Relevant Documents

- In this step we calculate similarity scores using both cosine similarity and Euclidean distance for both vectorization techniques (Binary and TF-IDF). We Retrieve the top 10 most relevant documents for each query.

5.2 Evaluation Metrics

- **Precision**: The ratio of relevant documents retrieved to the total documents retrieved.

$$\text{Precision} = \frac{\text{Number of Relevant Documents Retrieved}}{\text{Total Number of Documents Retrieved}}$$

- **Recall**: The ratio of relevant documents retrieved to the total relevant documents.

$$\text{Recall} = \frac{\text{Number of Relevant Documents Retrieved}}{\text{Total Number of Relevant Documents}}$$

- **F-Score**: The harmonic means of precision and recall.

$$\text{F-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

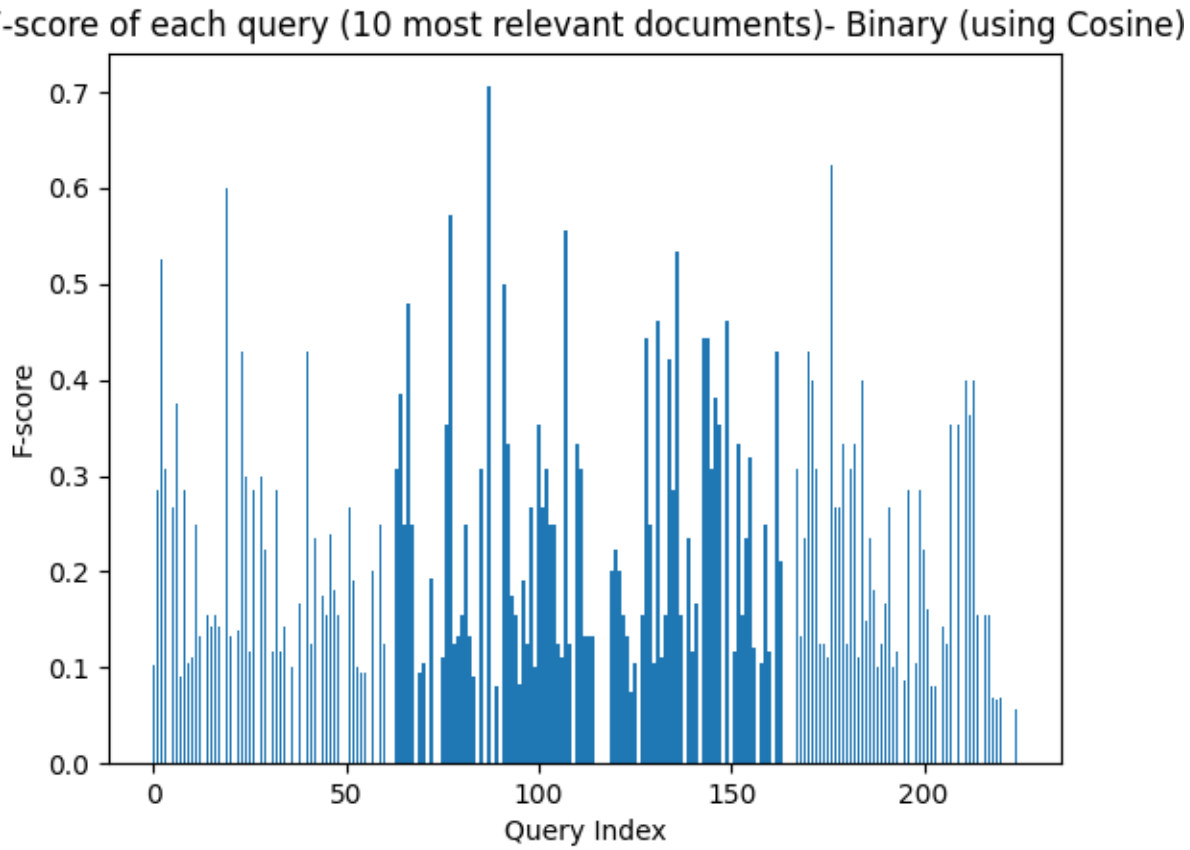
5.3 Calculation of Metrics

- **Precision, Recall, and F-Score**: We calculate these metrics for the top 10 documents retrieved for each query.
- Using this we can compare the performance across different vectorization techniques and similarity measures.

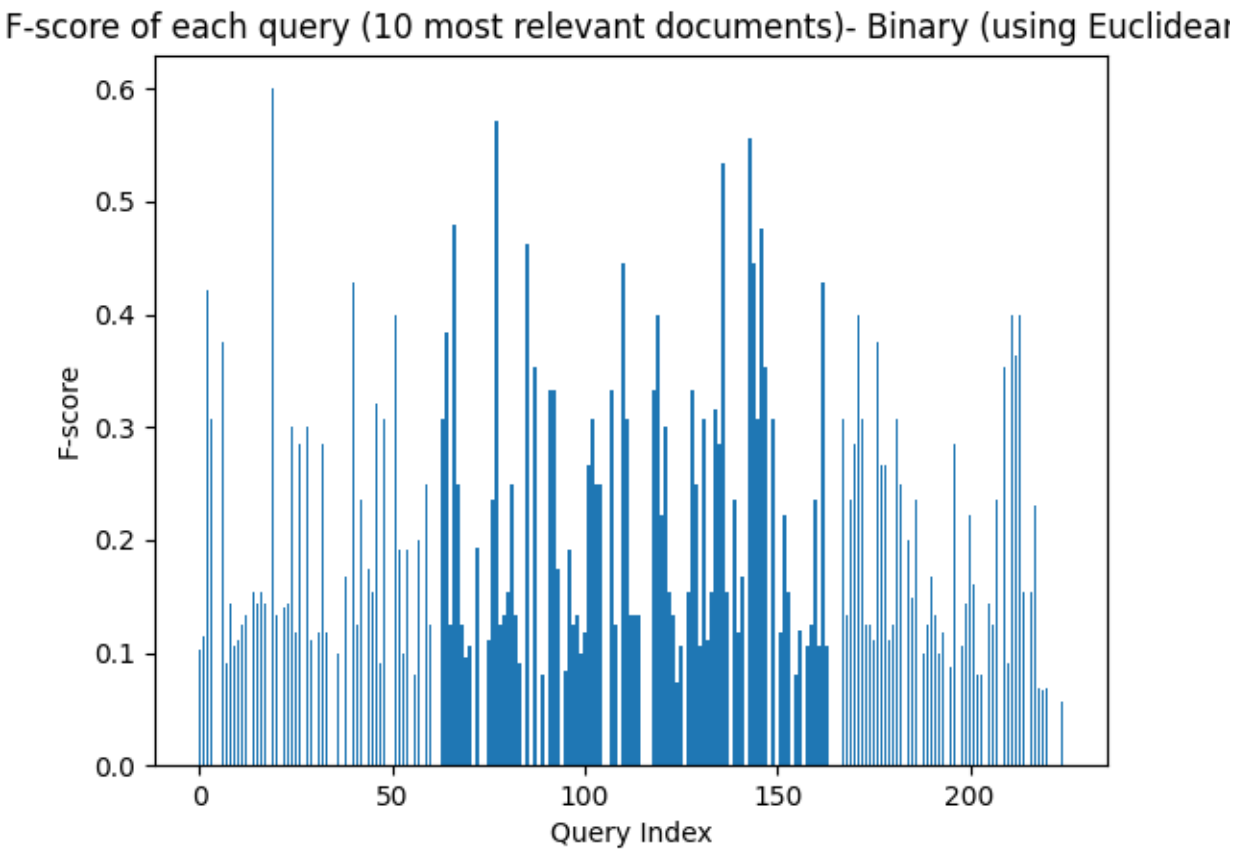
6. Visualization

- **Graphical Representation**: we use matplotlib.pyplot.bar to create bar charts for Precision, Recall and F-Score for each vectorization technique and similarity measure.
- **Saving Graphs**: We save each graph as '**{metric}_{vectorizer_type}_{distance_type}.png**' files.

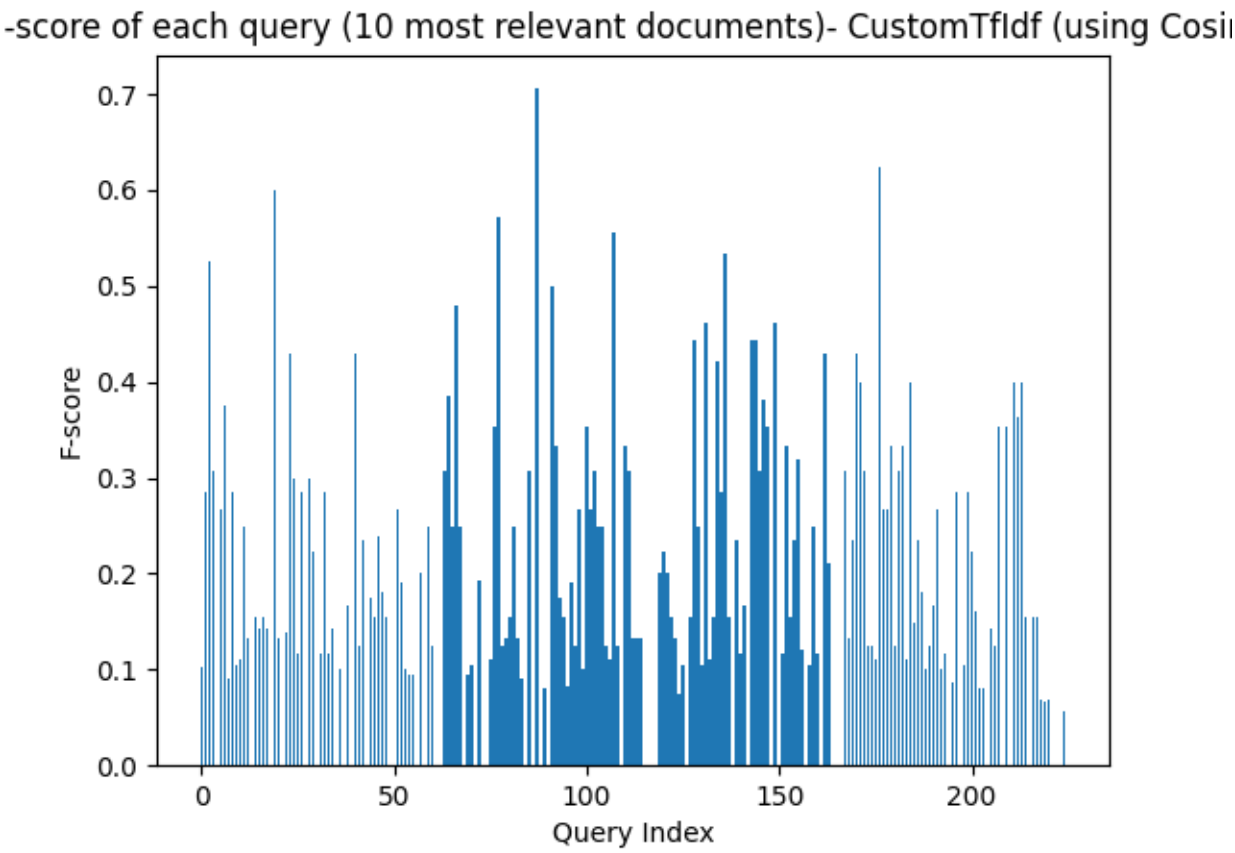
1.) F-score_Binary_Cosine



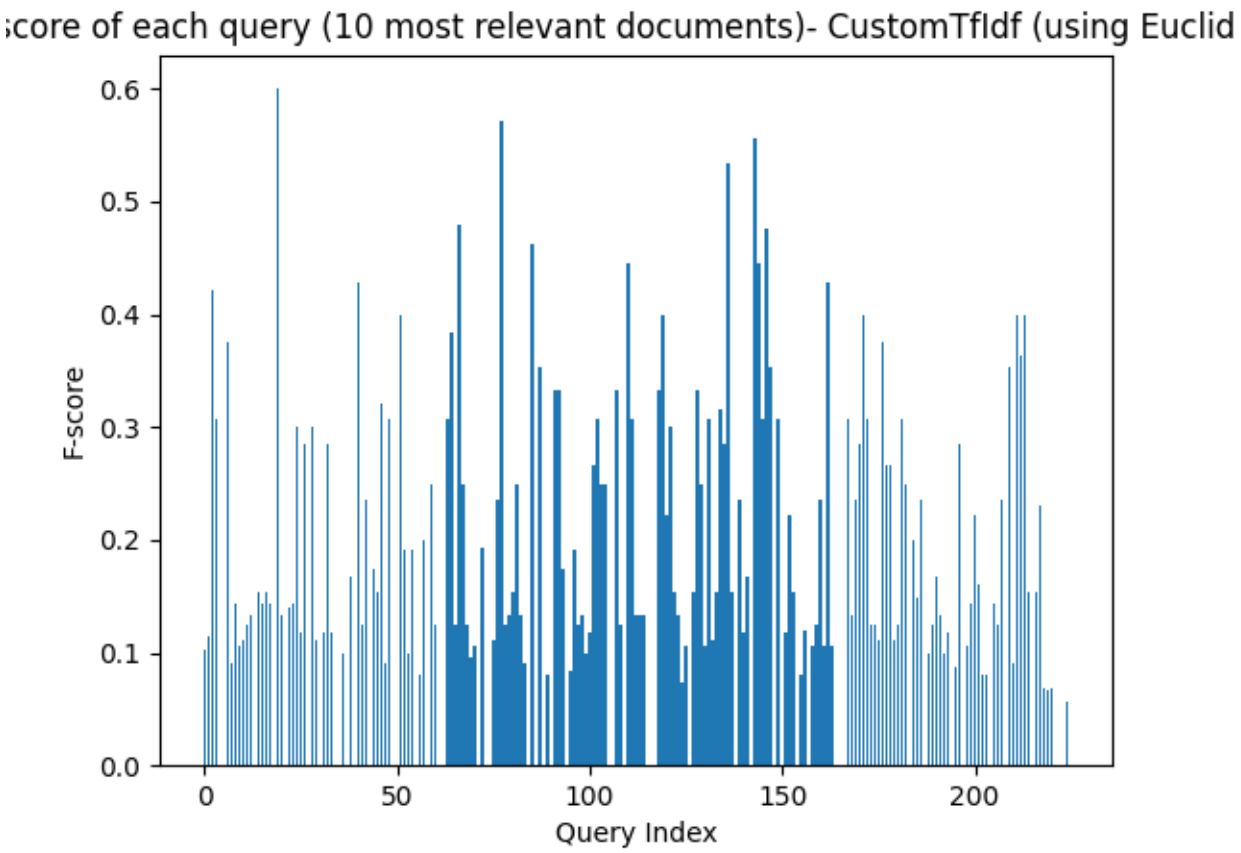
2.) F-score_Binary_Euclidean



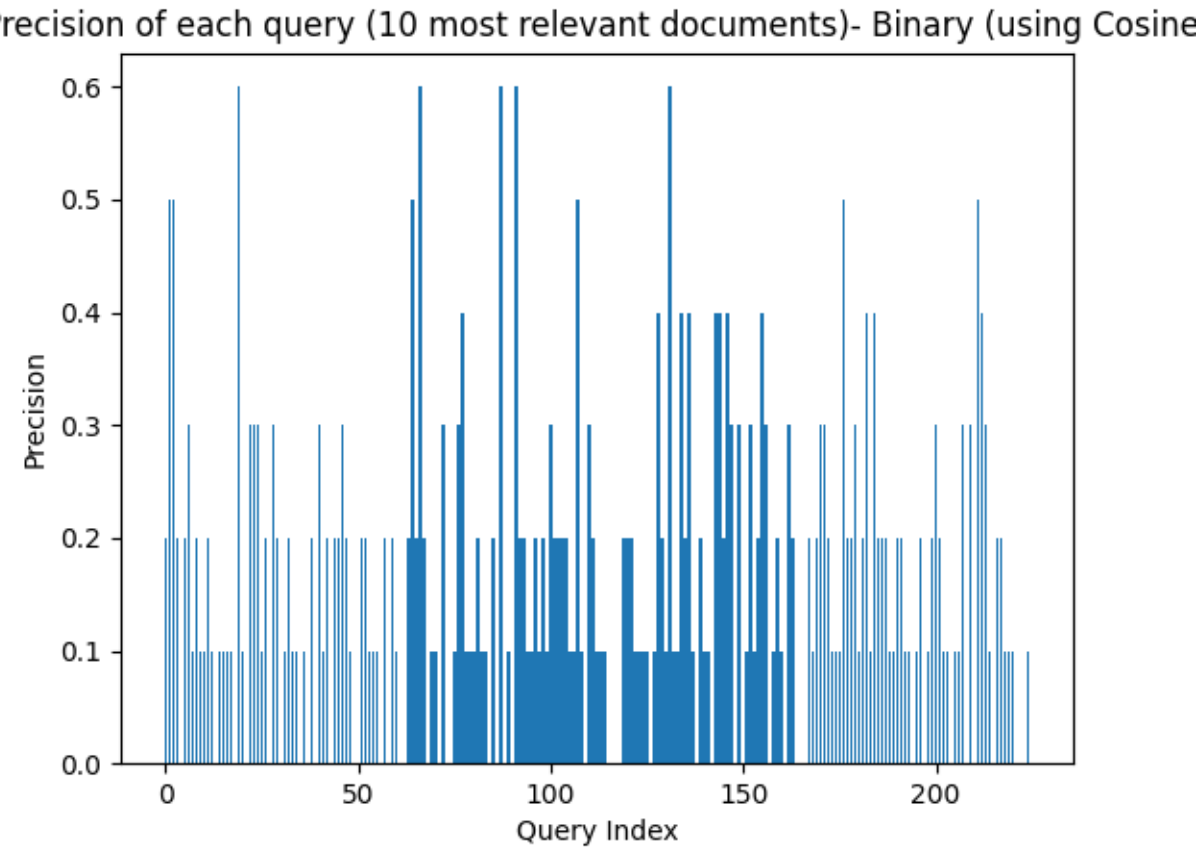
3.) F-score_CustomTfIdf_Cosine



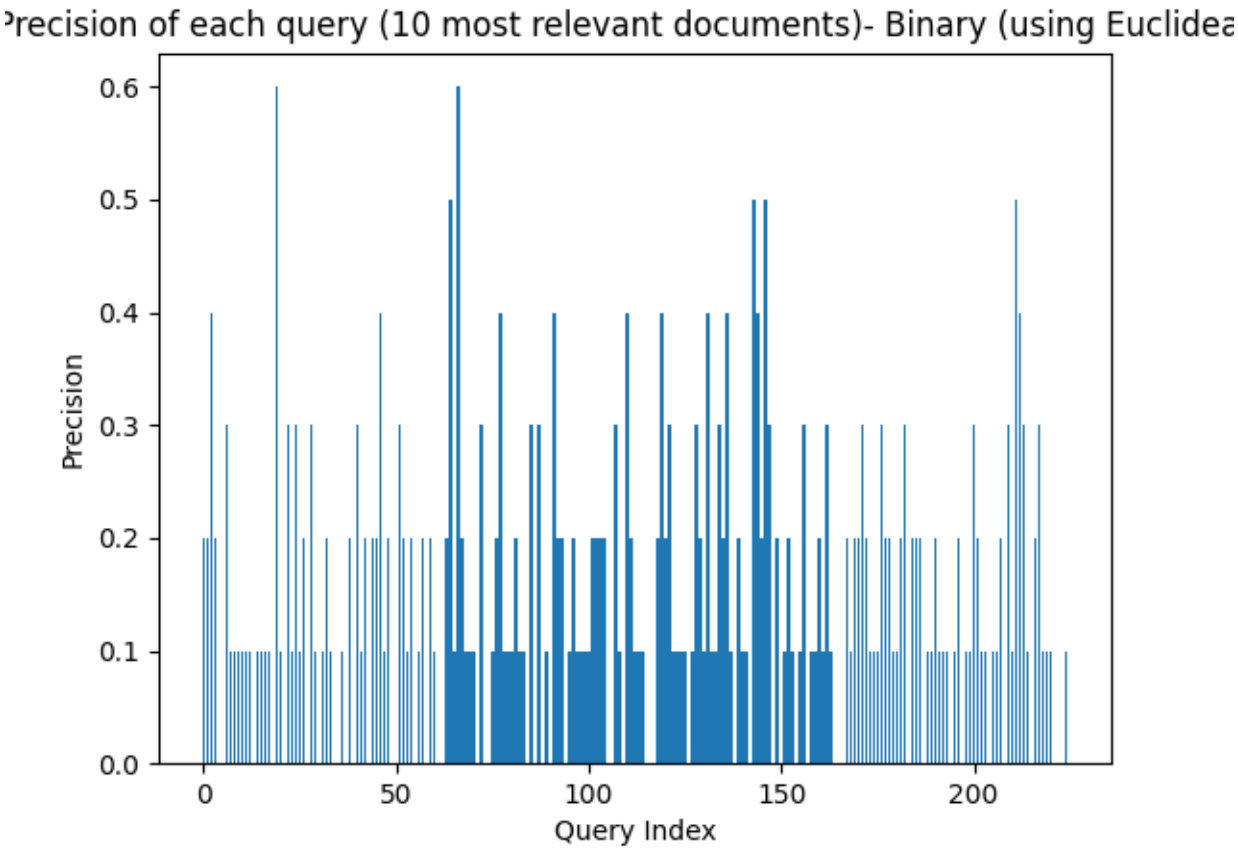
4.) F-score_CustomTfIdf_Euclidean



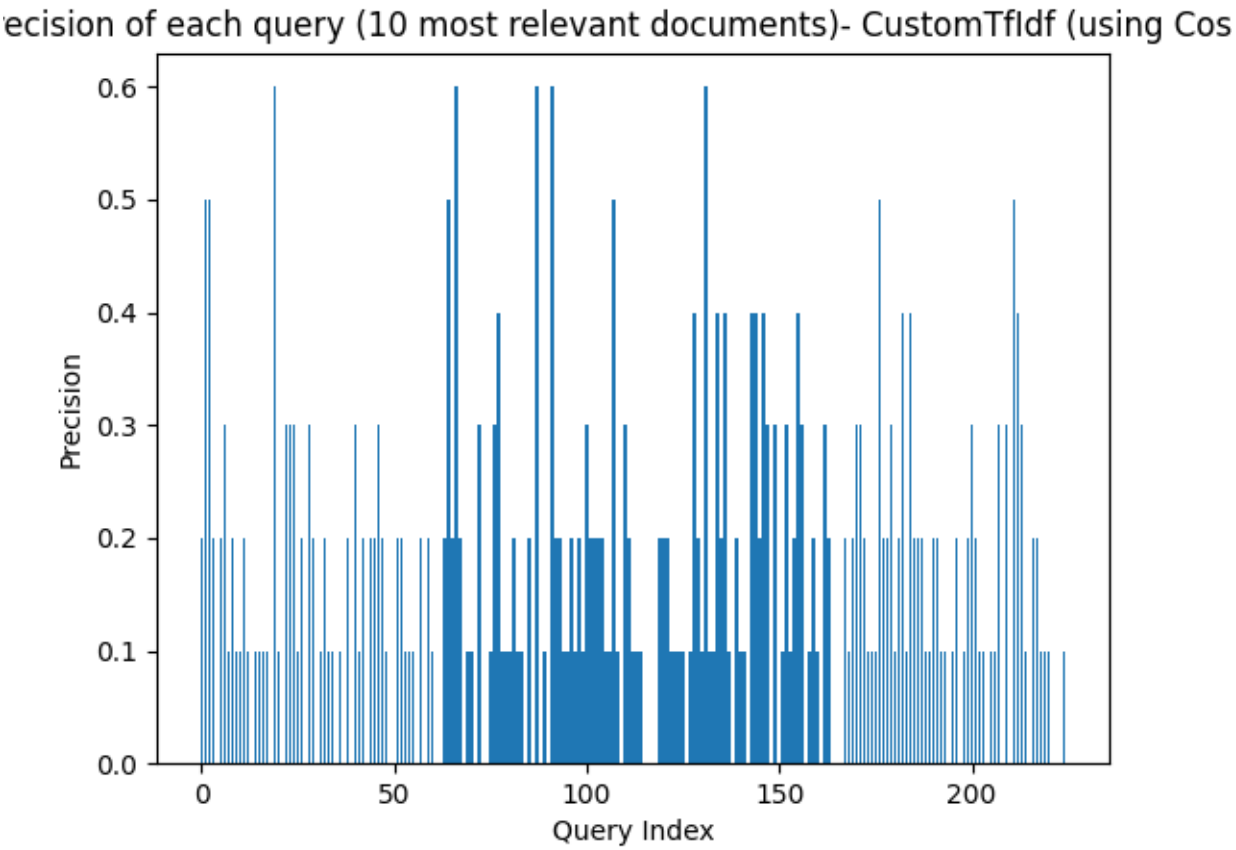
5.) Precision_Binary_Cosine



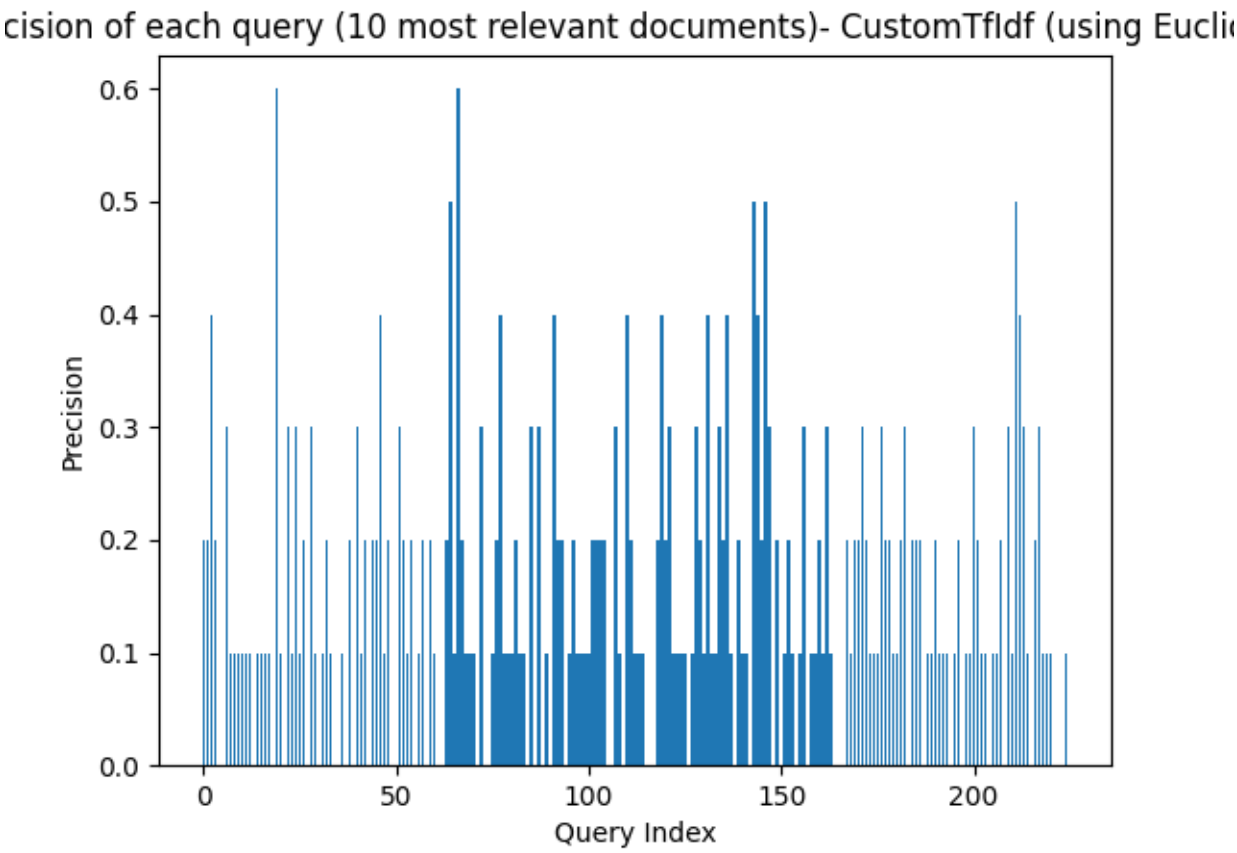
6.) Precision_Binary_Euclidean



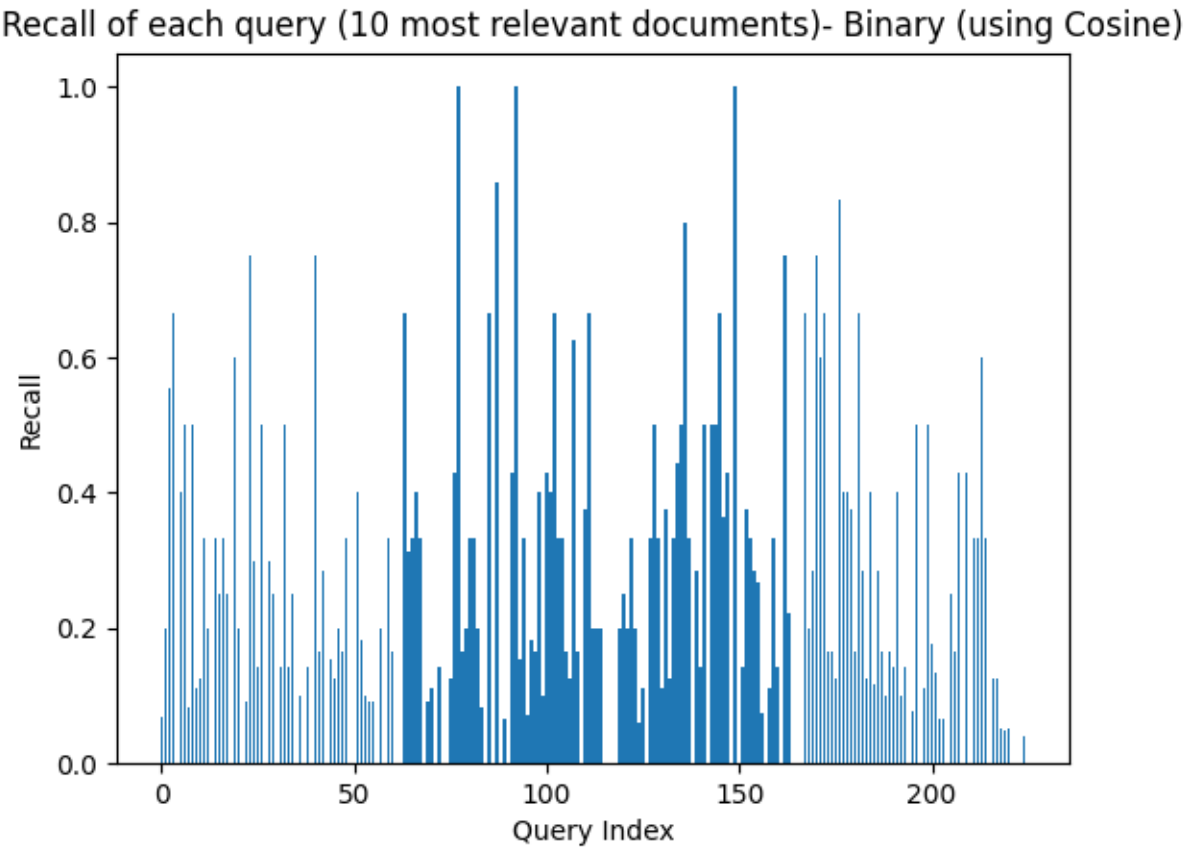
7.) Precision_CustomTfIdf_Cosine



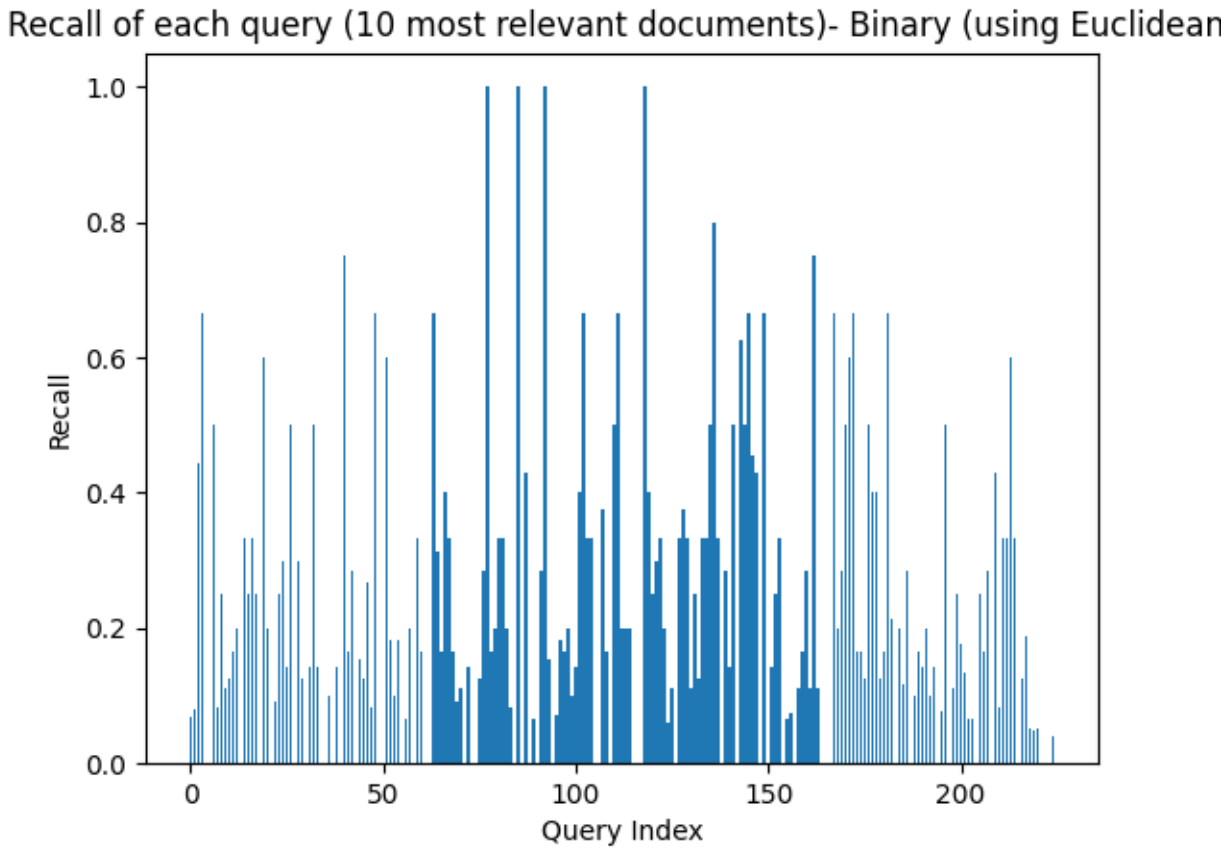
8.) Precision_CustomTfIdf_Euclidean



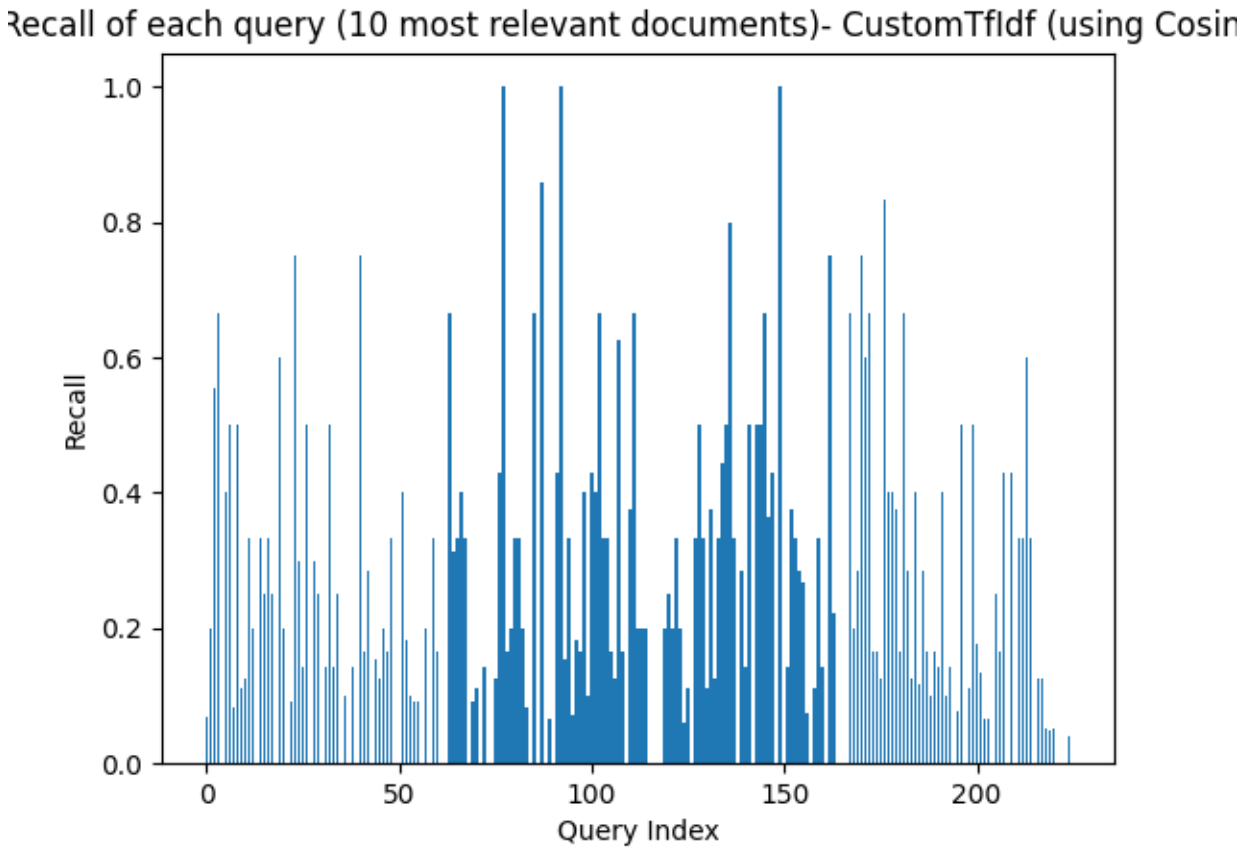
9.) Recall_Binary_Cosine



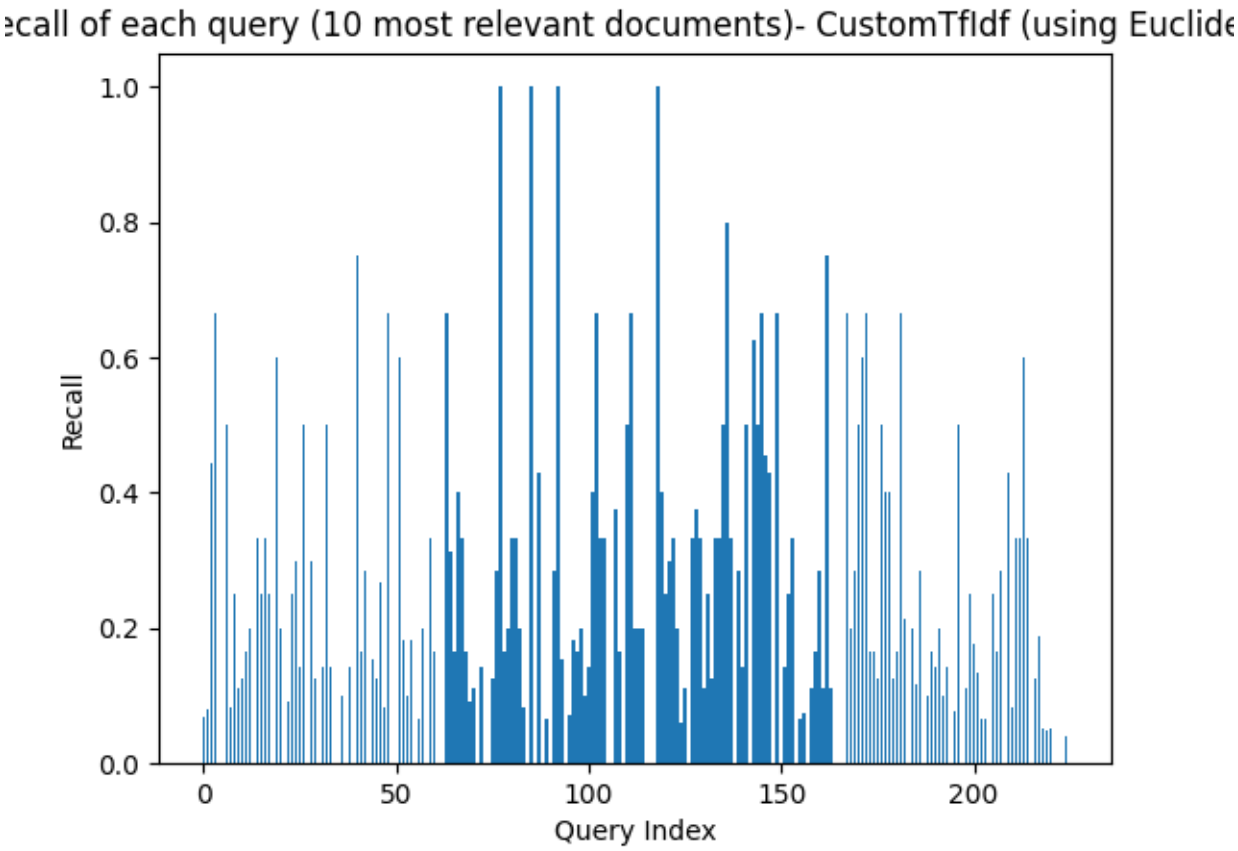
10.) Recall_Binary_Euclidean



11.) Recall_CustomTfIdf_Cosine



12.) Recall_CustomTfIdf_Euclidean



7. Results and Discussion

- **Mean and Maximum Scores**: We display the mean and maximum Precision, Recall and F-Scores in the terminal as per the specified format in the console as final output.

  <u>Console Output</u>:

```
C:\Users\hp\Desktop\Information Retrieval\Assignments\Assignement1IR\Assignment1InformationRetrieval>python assignment1.py
{'Binary': {'f': {'cos': (0.2684196452691116, 0.7058823529411764),
                  'euc': (0.0215344699034236, 0.26666666666666666)},
            'p': {'cos': (0.24355555555555555, 0.7),
                  'euc': (0.01955555555555556, 0.3)},
            'r': {'cos': (0.3533881875277164, 1.0),
                  'euc': (0.02845181978515312, 0.5)}},
 'TFIDF': {'f': {'cos': (0.17918355088418167, 0.7058823529411764),
                 'euc': (0.1612136761157517, 0.6)},
           'p': {'cos': (0.16088888888888891, 0.6),
                 'euc': (0.14444444444444443, 0.6)},
           'r': {'cos': (0.2420530682616765, 1.0),
                 'euc': (0.221552523127798, 1.0)}}}
```

- **Analysis**:

1. Variation Between Different Vector Models

→ **Binary Vectorizer vs. TF-IDF Vectorizer:**

Precision:

- The Binary Vectorizer may get higher precision while the presence of a term which is more indicative of relevance regardless of its frequency.

- The TF-IDF Vectorizer may get higher precision by giving more weight to rare but important terms which reduces the impact of common terms that appear in many documents.

Recall:

- The Binary Vectorizer may get lower recall because it does not consider term frequency which could be missing in documents where important terms appear frequently.

- The TF-IDF Vectorizer may enhance recall by taking that documents with frequently occurring relevant terms are retrieved.

F-Score:

- F-Score varies depending on the values of precision and recall for each vectorizer. TF-IDF often has given a better balance because it adjusts for term frequency and document rarity which potentially leading to higher F-Scores.

2. Variation Between Different Similarity Measures

→ **Cosine Similarity vs. Euclidean Distance:**

Precision:

- Cosine Similarity has higher precision because it normalizes the document length and focuses on the direction of the term vectors which makes it effective for identifying similar documents.

- Euclidean Distance might lower precision as it is more affected by the document length and absolute term frequencies which can introduce noise.

Recall:

- Cosine Similarity might have higher recall as it can effectively handle variations in document length and emphasize the importance of term co-occurrence patterns.

- Euclidean Distance may result in lower recall if the magnitude differences takeover the term matches.

F-Score:

- Similar to precision and recall the F-Score will likely be higher for Cosine Similarity in most cases due to its balanced approach to vector similarity.
- Euclidean Distance might not balance precision and recall which might result in lower F-Scores.

Effectiveness and Limitations of Similarity Measures and Vector Models

1. Similarity Measures

Cosine Similarity:
Effectiveness:

- It normalizes document lengths and focuses on the angle between vectors, which helps capture similarity in the presence and co-occurrence of terms.

- It's very Effective for documents of varying lengths and those where the relative frequency of terms matters more than their absolute counts.

Limitations:

- It's less effective for datasets where the magnitude of term frequency carries significant meaning.

- Can struggle with sparse datasets where term overlap is small.

Euclidean Distance:
Effectiveness:

- It's a straightforward and intuitive measure which is effective when the absolute differences in term frequencies are crucial for relevance.

Limitations:

- Sensitive to document length and term frequency magnitudes which leads to misleading similarity scores.

- Not ideal for high-dimensional and sparse datasets where distance can become less meaningful.

2. Vector Models

Binary Vectorizer:
Effectiveness:

- It's simple and computationally efficient which makes it suitable for initial filtering and scenarios where the presence of terms is sufficient to judge relevance.

Limitations:

- It ignores term frequencies which can lead to missing out on documents where term importance is frequency-dependent.

TF-IDF Vectorizer:
Effectiveness:

- Balances term frequency and document rarity which makes it effective for distinguishing important terms and reducing noise from common terms.
- It also adapts well to varied document lengths and provides more informative vector representations.

Limitations:

- More computationally intensive than the Binary Vectorizer.

- Can be less effective for very small or very large datasets where IDF values may not be reliable.

8. Conclusion

Overall, this assignment helps to apply the theoretical concepts learned during this course. The key things learned from the performance evaluation of the query processor and indexer developed for the Cranfield Dataset.