Implementing and Testing Relevance Feedback Methods: Analyzing Explicit Relevance Feedback Using the Cranfield Collection

Wilson Roldan

Abstract

This project investigates the impact of relevance feedback on information retrieval (IR) using the Cranfield Collection, a standardized dataset commonly used for IR experiments. The standard Rocchio algorithm was applied to adjust query vectors based on explicit relevance feedback, aiming to improve document retrieval accuracy. Preprocessed documents and query data were vectorized using TF-IDF, and cosine similarity was used to identify the top 5 relevant documents for each query. After applying the Rocchio algorithm, retrieval performance showed significant improvement, with higher precision, recall, F1 score, and Mean Average Precision (MAP). These results demonstrate the effectiveness of relevance feedback in refining retrieval results based on relevance judgments in the Cranfield Collection.

Introduction

Information retrieval (IR) is a critical area of research in computer science that focuses on the process of finding relevant documents from a large collection based on a user's query. Relevance feedback, which allows users to provide feedback on the relevance of documents retrieved by an initial search, is a well-known method for improving IR systems. The Cranfield Collection [1], a widely recognized dataset for IR experiments, serves as a standard test collection due to its predefined relevance judgments and structured format, allowing researchers to consistently evaluate and compare IR models. This project applies relevance feedback techniques, specifically the standard Rocchio algorithm, to improve the effectiveness of IR systems using the Cranfield Collection.

The Cranfield Collection contains 1400 aeronautics-related documents and 225 queries, with predefined relevance judgments for evaluating retrieval effectiveness. The dataset uses a five-level relevance scale to categorize documents based on their usefulness, ranging from no interest to a complete answer to the query. This graded relevance system provides more detailed feedback than binary relevance judgments, making it suitable for testing the impact of explicit relevance feedback methods like Rocchio on retrieval performance. The goal of this project is to apply the Rocchio algorithm [2] to adjust query vectors based on relevance feedback and assess the impact of this approach on retrieval accuracy.

The project involves several stages, including preprocessing the dataset, applying the Rocchio algorithm, and evaluating the retrieval performance using standard metrics such as precision, recall [3], F1 score, and Mean Average Precision (MAP) [4]. Initially, a baseline retrieval system is set up using TF-IDF [5] vectorization and cosine similarity [6]. The impact of relevance feedback is then measured by comparing retrieval results before and after applying the Rocchio algorithm. By leveraging the Cranfield Collection, this project demonstrates how relevance feedback can enhance the retrieval process by incorporating user-provided relevance judgments and improving the ranking of relevant documents.

Data and Methods

The Cranfield Collection is a popular and commonly used dataset for IR projects and experiments. It was created in the 1960's to study the effectiveness of IR systems. The dataset serves as a good example of a test collection because it has indexed documents, a set of test queries, and predefined relevance judgements. The Cranfield Collection is often referred to as a standardized dataset because of its well-defined structure that allows for consistent and reproducible experiments. Results from experiments on the Cranfield Collection are often comparable across studies due to its standard nature.

Cranfield Collection

The Cranfield Collection contains documents and queries related to the field of aeronautics. The version used in this project contains 1400 documents, 225 queries, and 1828 query relevance judgements. The Cranfield Collection uses a five-level relevance scale to categorize references based on their usefulness to a query. These levels are: (-1) no interest (12.2%), (1) minimum interest, often historical (7.0%), (2) useful for background or methods (21.1%), (3) highly relevant, critical to the research (40.0%), and (4) a complete answer to the question (19.8%) [7]. This graded relevance system provides nuanced feedback beyond binary judgments.

Preprocessing

The first step is to load the dataset using the ir_datasets library in Python and setting up tools for text preprocessing. To clean the text, a Porter Stemmer is initialized for stemming words and a set of English stopwords to filter out common, non-informative words. A preprocessing function is used to perform several tasks: it converts text to lowercase, removes punctuation, tokenizes sentences into individual words, and applies stopword removal and stemming. The result is a cleaned and standardized version of the text, which can then be used for tasks like calculating TF-IDF or applying relevance feedback methods.

Next, the preprocessing function is applied to both the documents and queries in the dataset. For the documents, the title, text, and author fields are combined into a single string, preprocess it, and store the cleaned version in a dictionary using the document ID as the key. The same approach for the queries, processing the query text and storing the results in another dictionary with query IDs as keys. These preprocessed dictionaries provide a structured and uniform representation of the data, which is essential for building a retrieval system and testing relevance feedback methods effectively.

Initial Retrieval

The first step is to transform the preprocessed text data into numerical representations for retrieval tasks. The TfidfVectorizer from the Sklearn library is used to convert the cleaned document and query texts into TF-IDF vectors. The TF-IDF representation helps highlight important terms in each document or query by assigning higher weights to terms that are unique to a particular text and lower weights to commonly occurring terms. First, the vectorizer is used to fit the document texts to learn the vocabulary and compute the document vectors. Then, the query texts are transformed into vectors using the same vectorizer to ensure consistency between the document and query representations.

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t.$$

Next, the similarity is calculated between each query and all documents using cosine similarity, which measures the angle between two vectors and is commonly used in information retrieval. For each query, the top 5 most similar documents based on their similarity scores are identified. These results are stored, including the query index, the indices of the top 5 documents, and their similarity scores, in a list. This structured output allows me to analyze the initial retrieval performance for each query and serves as the foundation for testing relevance feedback methods later in the process.

$$cos(\theta) = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|}$$

Rocchio Algorithm

The standard Rocchio algorithm is used to implement relevance feedback and explicit feedback. This is a well-known method for adjusting query vectors based on user feedback. The Rocchio algorithm modifies the original query by incorporating vectors from relevant and non-relevant documents, aiming to move the query closer to relevant documents and farther from non-relevant ones in the vector space. This approach effectively balances the original query intent with insights from user-provided relevance judgments.

$$Standard\_Rocchio: \quad \vec{q}_m = \alpha\,\vec{q} + \frac{\beta}{N_r}\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{N_n}\sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

To begin, it is necessary to define the parameters alpha, beta, and gamma, which control the weighting of the original query, relevant documents, and non-relevant documents, respectively. These parameters allow for fine-tuning the balance between maintaining the original query intent and incorporating feedback from relevance judgments. A dictionary is used to store relevance information for each query. This dictionary maps each query ID to a list of associated documents, where each document entry includes the document ID and its relevance score. The dictionary is populated by iterating through the dataset's relevance judgements, which serves as the foundation for relevance feedback.

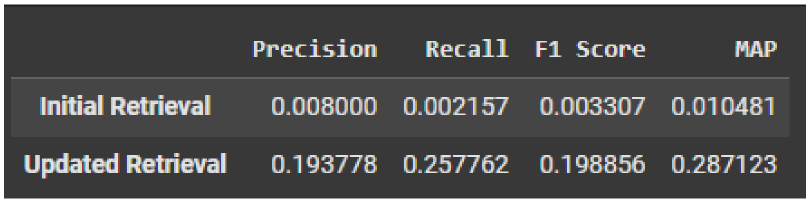
Next, each query is processed to compute updated query vectors using the Rocchio algorithm. For each query, it is important to first check if relevance judgments are available in the dictionary that holds the relevance data. If no relevance data is found, the Rocchio update is ignored, and the original query vector is retained. Otherwise, the associated documents are classified into two categories: relevant and non-relevant. For each relevant and non-relevant document, the vector representation is retrieved and converted to a dense format. These vectors are then stored in lists, which are later used to compute the feedback components of the updated query vector.

With the relevant and non-relevant documents identified, the Rocchio update is calculated. The updated query vector is computed by combining the original query vector (scaled by alpha), the mean vector of relevant documents (scaled by beta), and the mean vector of non-relevant documents (scaled by gamma). Instances where there are no relevant or non-relevant documents are skipped and do not contribute to the updated query vector. The resulting vector is reshaped into a two-dimensional format to ensure compatibility with subsequent processing. Each updated query vector is then appended to a list, which holds all modified query representations.

Finally, the retrieval process is re-run using the updated query vectors to observe the impact of relevance feedback. The cosine similarity is calculated for each updated query vector, with all document vectors to determine the most relevant documents. I extract the top 5 documents with the highest similarity scores and store these results, including the query index, document indices, and similarity scores. This step completes the feedback loop, enabling the system to refine its retrieval results based on relevance judgments and potentially improve precision and recall in subsequent evaluations.

Evaluation

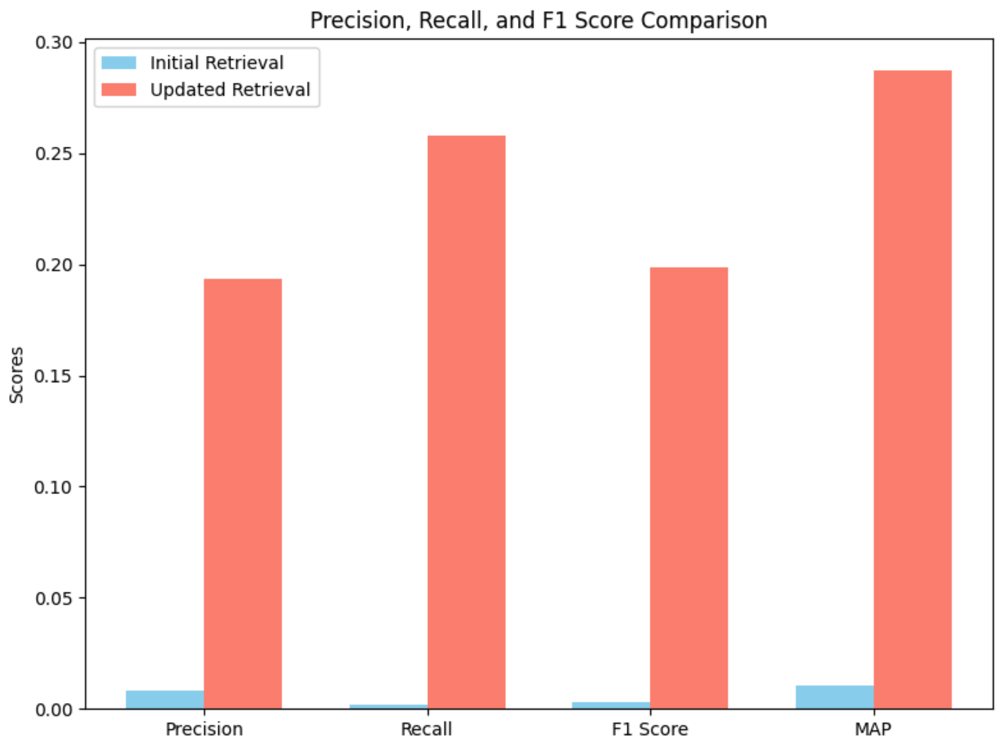
The retrieval performance is evaluated by computing precision, recall, and F1 score for both the initial and updated retrievals. For each query, the set of relevant documents from the corpus and compare it with the set of retrieved documents. Precision, recall, and F1 score are determined by calculating the overlap between these two sets. Precision is the proportion of retrieved documents that are relevant. Recall is the proportion of relevant documents that are retrieved. The F1 score is the harmonic mean of precision and recall for each query. These metrics are averaged across all queries to produce the final evaluation scores for the retrieval process. Mean Average Precision (MAP) is also calculated in order to provide a more detailed evaluation of retrieval performance. MAP calculates the precision at each rank where a relevant document appears and averages these values across all ranks for a query.



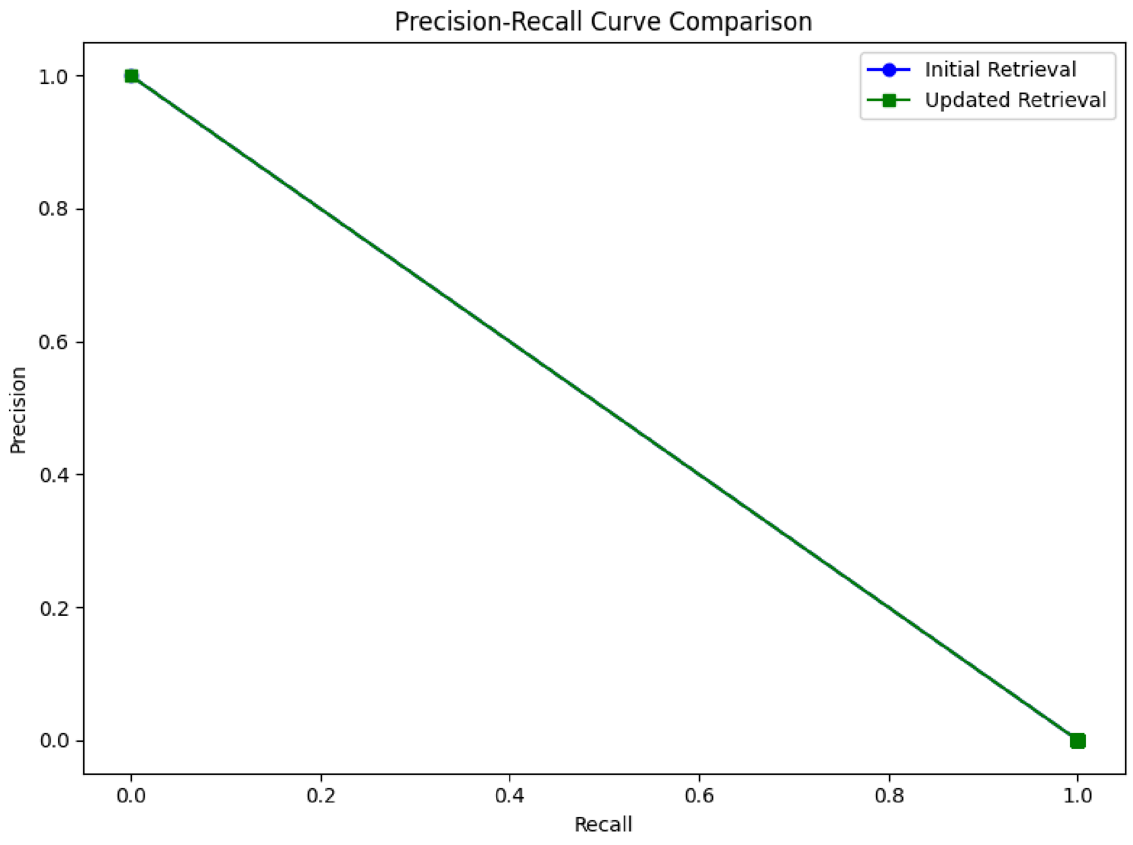
	Precision	Recall	F1 Score	MAP
Initial Retrieval	0.008000	0.002157	0.003307	0.010481
Updated Retrieval	0.193778	0.257762	0.198856	0.287123

Results

The Rocchio algorithm had a notable impact on retrieval performance when comparing the updated retrieval metrics to the initial retrieval metrics. Initial retrieval showed low precision (0.0080), recall (0.0021), and F1 score (0.0033) with MAP 0.010481. The updated retrieval, after applying the Rocchio algorithm, showed higher precision (0.1937), recall (0.2577), and F1 score (0.1988) with MAP 0.287123. The retrieval metrics have significantly improved, indicating that the relevance feedback has been successfully applied, and the ranking of documents has been reordered to prioritize relevant documents.

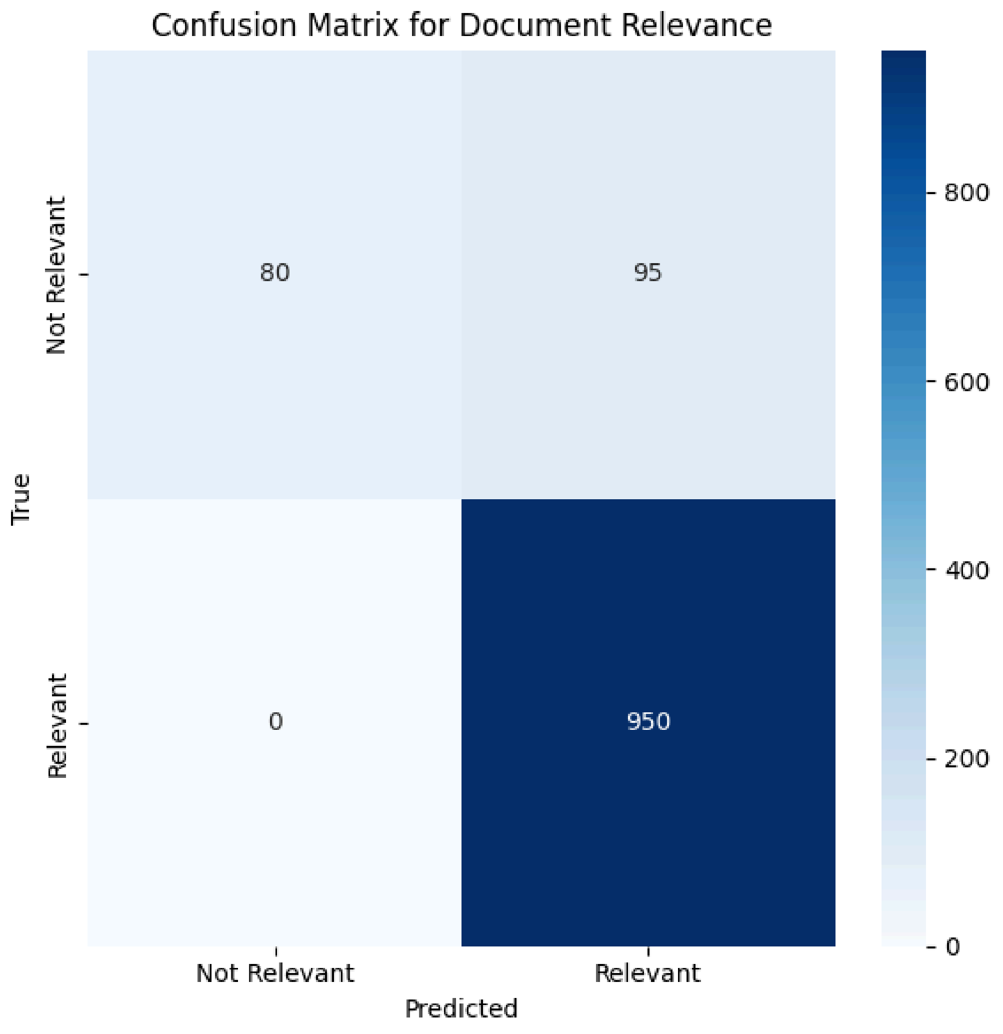


Limitations of the dataset can be observed by comparing different metrics. The dataset is relatively small, and each of the queries has few relevant documents. The precision-recall curve can demonstrate how this becomes a problem. The low recall at first suggested that the model only retrieved top-ranked relevant documents. The model was accurate at this stage and precision is high. Precision can be high with low recall because the model retrieves only a small number of relevant documents from the larger collection. The precision then begins to decline, while the recall increases. This is because the model has retrieved all relevant documents early and is now increasing the corpus size with only non-relevant documents.



The confusion matrix illustrates how the documents were classified by the model based on relevance. The model correctly identified 80 documents as ’Not Relevant’. The model incorrectly identified 95 ’Not Relevant’ documents as ’Relevant’. The model correctly identified

950 documents as ’Relevant’ and there were no cases where relevant documents were misclassified as ’Not Relevant’. The model is highly effective at identifying relevant documents but could improve in minimizing the misclassification of irrelevant documents as relevant (reduce false positives).



Conclusion

In this project, methods for relevance feedback were implemented and tested, with the standard Rocchio algorithm applied to improve information retrieval results. Using the Cranfield collection, a widely recognized and standardized dataset for evaluating information retrieval models, the impact of explicit relevance feedback on retrieval performance was assessed. Evaluation metrics, including precision, recall, F1 score, and Mean Average Precision (MAP), were used to measure the effectiveness of both the initial retrievals and those updated through relevance feedback. The results demonstrate how relevance feedback, as implemented through the Rocchio algorithm, can refine the retrieval process and improve the accuracy of document rankings based on relevance judgments from the Cranfield collection.

References

- [1] "Cranfield experiments," *Wikipedia*, [Online]. Available: https://en.wikipedia.org/wiki/Cranfield_experiments. [Accessed: Dec. 20, 2024].
- [2] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, 2nd ed. Boston: Addison-Wesley, 2011, p. 20.
- [3] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, 2nd ed. Boston: Addison-Wesley, 2011, pp. 11-22.
- [4] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, 2nd ed. Boston: Addison-Wesley, 2011, p. 27.
- [5] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, 2nd ed. Boston: Addison-Wesley, 2011, pp. 32-48.
- [6] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, 2nd ed. Boston: Addison-Wesley, 2011, p. 57.
- [7] "Cranfield dataset," IR Datasets, [Online]. Available: https://ir-datasets.com/cranfield.html. [Accessed: Dec. 20, 2024].