RelEx - Relevance Feedback with XAI Features

Govind Shukla*, Pritha Ghosal*
*Otto-von-Guericke University, Magdeburg

***Abstract*—Relevance feedback in search engines is well understood as a method that helps generating more relevant results by refining query based on user's preferences and information need. However, users lack knowledge and contextual awareness which can impact the effectiveness of search results. Moreover, some users also want to see how their feedback affects their results. Therefore, in this project we built RelEx, an XAI exploratory search engine that takes explicit feedback from users in an interactive way and then tune the search results using the modified query and contextual information. Following the traditional Relevance Feedback system that uses the conventional query modification algorithm i.e., Rocchio algorithm, our explainable relevance ranker extended Rocchio algorithm by adding weighted key-phrase vectors. Subsequently, we attempt to include user feedback that contributes towards finding a feedback score that constitutes the cosine similarity of the modified query & key-phrase similarity. In an online user study, we assess the proposed interface and ask participants to rate its explainability and assessability. According to the evaluation results, RelEx demonstrates its potential to actively include the user in the decision-making process of the search engine and make it straightforward and transparent to them. The report concludes that relevance feedback with explainable elements can help users make more informed decisions and better utilize search results.**

***Index Terms*—Explainable Rankings, XIR, XAI, Re-ranking, Search engines**

I. Introduction

In large communities with a broad extent of digitized information, it is often important that the user can not only identify relevant documents but also understand the significance of the rankings. Arguably this relevance has a fair share of the semantic and syntactic notions of textual similarity. Therefore, in addition to the ad-hoc information retrieval system, the scientific community is increasingly interested in systems capable of explaining the reasons behind the rankings in terms that are comprehensible to a non-expert. This has surged the demand for extracting explanations from such black-box (Artificial Intelligence) models.

While relevance feedback can improve the accuracy of search results, it does not provide users with insight into the reasoning, why a document is relevant to a query. The developer of a system might be able to interpret how the system works and predict how it would perform in unforeseen events. This lack of transparency can result in user frustration and a lack of trust in the system. To overcome these limitations, the project aims to develop RelEx, a prototype of user-interactive XAI search system with relevance feedback. The user feedback is further used to refine the ranking of search results along with explainable elements such as key-phrases and context-matching of the selected relevant documents. By doing so, users can better understand how the system arrived at its results, which can improve their trust and faith in the system. Most users have a general assumption of which documents should be selected as relevant. However, users lack the knowledge or contextual understanding to formulate their queries. Relevance feedback method like Rocchio algorithm are used to enhance the queries according to user needs. The motivation behind the project is to address the limitations of traditional relevance feedback techniques in information retrieval systems and improve their effectiveness and enhance the user experience. Mi and Jiang et al., 2019 [1] used key-phrase and summaries as their elements of explanations and concluded that the search result summary plays a crucial role in explaining the retrieval. Following their work we reach our research questions, which are the following:

- **[RQ1]** How well do the key-phrases in comparison to the summary of a document help users select relevant documents during Relevance Feedback? (Assessability)
- **[RQ2]** How do users rate the explanations in our explainable interface in comparison to a standard interface, after Relevance Feedback? (Explainability)
- **[RQ3]** Does the extended Rocchio Algorithm provide the user with more pertinent results than the original Rocchio Algorithm?

In order to support these research questions, we target the following modifications -

1) **In re-rank with Relevance Feedback approach**: The user's search query is enhanced to contain an arbitrary assortment of relevant and irrelevant documents. Firstly, our inclusion of highlighted key-phrases aims to aid the user in making the selection of relevant documents. Secondly, along with the relevant and non-relevant document vectors we have also included the extracted key-phrase vector of those documents as an additional term in the equation of Rocchio to calculate the modified query. Also, extracting context for the key-phrases helps in grasping the general content and context of the document. Lastly, the feedback score of each document constitutes a combination of the cosine similarity between the modified query and document vector, and the matching-context count. We believe that this potentially increases the ranking of relevant documents and decreases the ranking of irrelevant documents, resulting in more relevant documents in the top 10 retrieved results.
2) **Explainable elements**: Our approach focuses on multiple constituents of explainability on two levels -
    - *Global level:* Shown at the overall feedback-result level

  - Graphical representation of Query vector modification.
- *Local level:* Shown at each document level
  - Highlighted Key-Phrases
  - Highlighted context-match
  - Bar chart for context-match count

Moreover, to facilitate replication and reuse of our work, we have made the implementation of our back-end code[1] and front-end code[2] on GitHub publicly available.

The remainder of this paper is organized as follows: The following section (section II) describes the concepts and technologies closely coupled with our experiment and then we highlight the current state-of-the-art research in Section III. Next, In part IV, we lay out our contribution to the ranking approach and methodological explanation and outline the architecture and implementation details of RelEx alongside our choices of visualization of the explanations. Section V contains an elaborate working mechanism of the prototype. Finally, section VI & VII explains our approach to evaluate the ranking system with a benchmark dataset, and the challenges we faced. In section VIII, we finally conclude the explainability and assessability results of our system.

II. Background

This section covers the supporting concepts or research for our project such as Learning to rank (L2R), relevance feedback, and Rocchio algorithm.

A. Learning to rank (L2R)

The process of Learning to rank (L2R) for information retrieval (IR) involves automatically building a ranking model using training data so that the model can classify forthcoming items based on their relevance, significance, or preferences. Here "Ranking" refers to the process of sorting documents according to relevance in order to locate information relevant to a query[2]. A base machine learning model is used in the training phase to predict a relevance score s_i = f(x_i) against a relevance score y_i, where $\vec{x}$ is considered to be the input query-document pair i.e., $\vec{x_i}$ = ($\vec{q}$, $\vec{d_i}$). Based on the loss computation L2R models are categorized into Pointwise, Pairwise & List-wise approaches. Prior to experimenting with the performance and effectiveness of a few other neural rankers on our TREC-COVID-19 dataset, we concluded that the pointwise ranker MonoT5 outperforms the other ones based on some metrics of retrieval quality (as described in Section IV-C & Table I).

B. Relevance feedback

Relevance feedback is the approach of achieving a more relevant set of search results depending on the user's preferences and judgments. Adjusting to the user's needs and context, the system can increase the accuracy and relevancy of the information retrieval. According to Chapter 9 of "An Introduction to Information Retrieval" by Manning, Raghavan & Schütze et al., 2009[3] the fundamental procedure of acquiring feedback is to return an initial set of retrieval results based on the user's (short, simple) query. The user then selects some returned documents as relevant or non-relevant. Based on user feedback, the system computes an improved illustration of the information need. An updated set of retrieval results are shown by the system. There are two basic categories of relevance feedback [4], [5]:

- *Explicit:* In this case the user has to explicitly declare whether documents are pertinent to or unrelated to their query, typically by rating, ranking, or selecting them.
- *Implicit:* Here the system attempts to estimate the documents that might interest the user, inferring from their actions, such as clicking, viewing, or scrolling.

For our prototype, we have decided to leverage explicit relevance feedback. There are some baseline condition that needs to be abide by in this approach i.e., firstly the user must possess the necessary knowledge to be able to formulate an initial query that is at least somewhat relevant to the contents they seek. Secondly, the term distribution of relevant documents will be semantically and syntactically similar to the ones that the user selected and the terms of the non-relevant documents would not match those. This brings us to the next focus of our approach which is extracting key-phrases and finding resemblance with other documents.

C. Rocchio algorithm

Rocchio algorithm is a method of relevance feedback that enhances the user's query to increase the recall and precision in information retrieval systems. The concept was first introduced by J.Rocchio in 1971 [6] is one of the most popular implementations of relevance feedback. The Rocchio algorithm modifies the query vector by adding a weighted combination of the vectors of the relevant and irrelevant documents to the original query vector. The weights used for the relevant and irrelevant documents are determined by three factors: alpha, beta, and gamma. The equation[7] of modified query vector $\vec{Q}_m$ is given by

$$\vec{Q}_m = \alpha\vec{Q}_0 + \frac{\beta}{|D_r|}\sum_{\vec{D}_j \in D_r}\vec{D}_j - \frac{\gamma}{|D_{nr}|}\sum_{\vec{D}_k \in D_{nr}}\vec{D}_k \qquad (1)$$

where $\vec{Q}_0$ is the original query vector D_r and D_{nr} are sets of vectors containing co-ordinates of relevant and non-relevant documents. Generally, the hyper-parameters α is set to 1, β to 0.8 and γ to 0.3.

III. Related Works

We have referenced a few earlier attempts of explaining relevance feedback to understand how our project differs from the existing array of research in this field. After the initial review of the abstracts, research questions, and methodologies of these papers, we identified a few that are the most closely related to our research. One of the earliest works done by Mi and Jiang et al., 2019 [1] precisely points out the motive behind explainable approaches. The authors state that low

[1]https://github.com/govind17/Information-Retrieval-Project

[2]https://github.com/ghoPritha/DataVisualizer-XAI-relevance-feedback

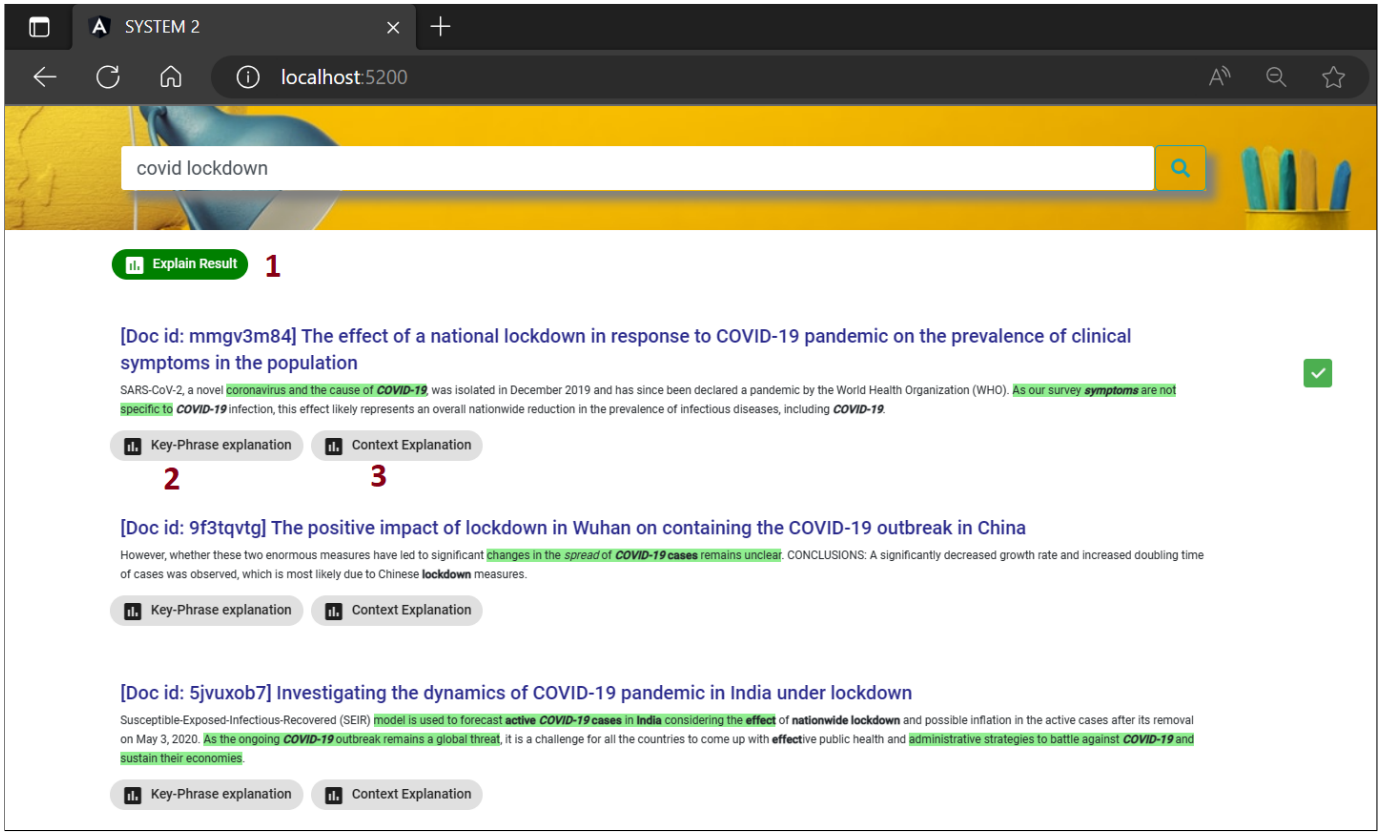

Fig. 1. This is the RelEx application interface post Feedback. Showing the result of Relevance Feedback for the query 'Pandemic death'. The component marked by '1' explains the Rocchio classification shown in Fig:4. Component '2' referred to as 'Key-phrase explanation' shows the matching key-phrase with 'chosen relevant document' as shown in Fig:3. Component '3' is explained in Fig:2

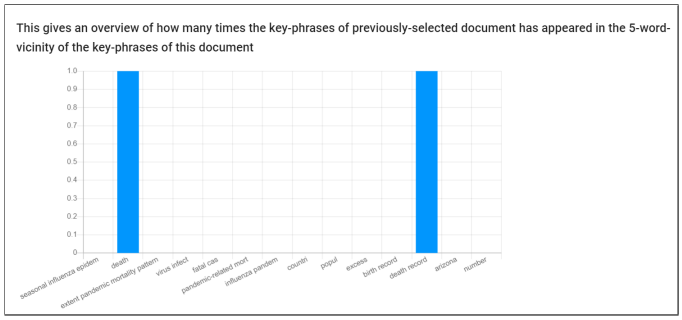

Fig. 2. *Context Explanation*

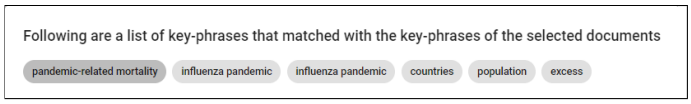

Fig. 3. *Key-phrase Explanation*

transparency means users do not understand the retrieved results and low assessability means that the summary did not assess the user if the result would be useful, leading to fewer clicks on a result. As a measure of attractiveness, they used key-phrase matching and document summaries. And by judging the summary and result during the user study they concluded that the search result summary plays a crucial role in explaining the retrieval of a particular result and the usefulness of the result and its comprehensibility heavily influence users' click decisions. Following their work, Chios and Verberne et al., 2021[8] address the explainability task on the query level and on the document level. They employed DRMM to deduce an explainable user interface where the query term relevance is viewed through a doughnut chart and a passage ranker to determine the relevant passage and visualize it as a document thumbnail with passage highlights. They also stated that a personalized search engine should be sufficiently transparent for two main reasons - **1.** to establish trust with the user by assuring them that personalization does not compromise the quality of results, even if other users receive different outcomes. **2.** to provide transparency to the user regarding the personal information utilized in result ranking. We also closely examined ExS by Singh & Anand et al., 2022, reference Haque2022, and ExDocS, cite Pollley2021, and discovered that their question of focus is the most similar to ours. ExS is a modern baseline explainable search system (EXS) that uses the Deep Relevance Matching Mode (DRMM), a pointwise neural ranker, to determine relevance. It also uses the LIME posthoc explanation technique (Ribeiro 2016) to show how such a system can successfully aid a user in understanding the results of neural rankers and highlighting areas for de-

velopment. ExDocS[9] on the other hand, is based on a re-ranking of comprehensible components of evidence, such as term statistics, contextual words, and popularity as indicated by citations. They include locally comprehensible elements like textual justifications, "math behind the rank" for each re-covered text, and an extensive defense in the form of a side-by-side comparison of various acquired papers. Now, inspired by the other works, we explored methods for narrowing the gap between the model's perspective and the user's perspective that would also take the user's feedback into the rank determination of documents. We introduce additional terms in the standard Rocchio algorithm to enhance the performance of relevance feedback and explainable features to provide explanations in a comprehensible, and visually appealing way.

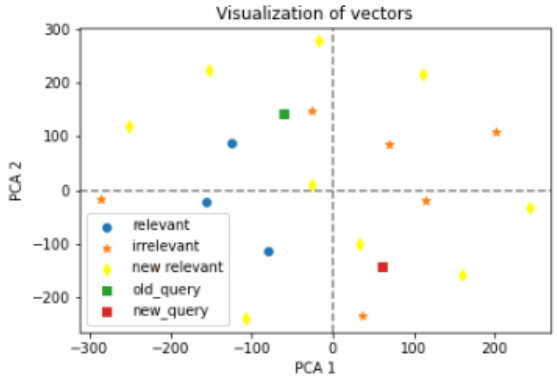

Fig. 4. Rocchio classification graph: The blue points are relevant documents selected by the user, orange star points are irrelevant documents that are not selected documents among the top-10 results by the user, yellow diamond points are documents that are newly added after the calculation of feedback score 2, finally, green and red box points are old and new query respectively

IV. Methodology

Fig:5 gives an overview of our methodology, which consists of data pre-processing, XAI feature extraction, ranking of text, collecting feedback from users, re-ranking the documents with relevance feedback, and generating explanations. In the feedback step, the user inputs are collected as a set of relevant documents and irrelevant documents. This input from the user helps in enriching the query through our enhanced Rocchio algorithm 2 as well as collecting key-phrases from relevant documents for further relevance ranking.

A. Data Pre-processing

This step prepares data ready for search results and ensures there are no duplicates or empty text in the result page. We used TREC-COVID-19 Dataset [10][11] for our experiments which is a collection of scholarly articles, pre-prints, and other research-related documents that have been made available in response to the COVID-19 pandemic. The dataset also contains relevance judgments for a set of queries. Therefore it is widely used by researchers to evaluate and compare different retrieval models, ranking algorithms, and ad hoc search strategies. For our purpose, we extracted the following fields only -

- **docno:** Indicates a unique number assigned to each document.
- **title:** Title of text.
- **abstract:** A summary of the document.

Out of *192,509* records, we found *400* duplicates, and *54,621* records that contain null values that were later removed. This sanctified the search result which is shown to the user in the front-end. Furthermore, we used PyTerrier Python library to build an index of the 'abstract' field and the 'docno' as metadata. The indexes enable fast and efficient search and retrieval of relevant documents by creating a data structure that maps terms in the documents to their locations in the documents.

B. XAI feature extraction

In this step, we extract hand-crafted facets which are capable of explaining re-rank rationale (XAI or Explainable Artificial Intelligence), i.e., summary and key-phrases, from the text. These features also help the user to decide which documents are relevant on the resultant page when the user sees the result initially returned from the query search.

Summarizing text: We summarized the abstract text in *2 sentences*. We used LSASummarizer as a summarization algorithm implemented in the Python library Sumy. It uses Latent Semantic Analysis(LSA) to summarize the text. LSA is a technique in natural language processing that analyzes relationships between a set of documents and the terms they contain. The LSA algorithm converts the text into a mathematical representation and identifies important topics in the text. The abstract is first tokenized and parsed using the Tokenizer and PlaintextParser classes from the Sumy library. Then, an LSASummarizer item is created with default parameters, and the number of sentences to output is set to two. The summarizer is then run at the parsed document, and a summary of the specified length is returned. Fig:6 illustrates an example of summarization of a text using LSASummarizer summarizing the text into two sentences.

Key-phrasification: Key-phrasification is the process of extracting a set of phrases that captures the essence of the document. Extracted key-phrases contain one word to four words. As key-phrases distill the important information from documents, they are useful for applications such as information retrieval and query expansion. Key-phrases help improve document retrieval efficiency significantly as proved by Savary et al., 2020 [12]. We used TopicRank Algorithm for key-phrase extraction on the 'abstract' field, which identifies the longest sequences of nouns and adjectives as key-phrase candidates. Next, it clusters these candidates into topics, constructs a graph of topics, and ranks the topics using a random walk algorithm to determine the final set of key-phrases. Finally, we collected them in a list and stored them. An example of key-phrasification by using a python based toolkit available on

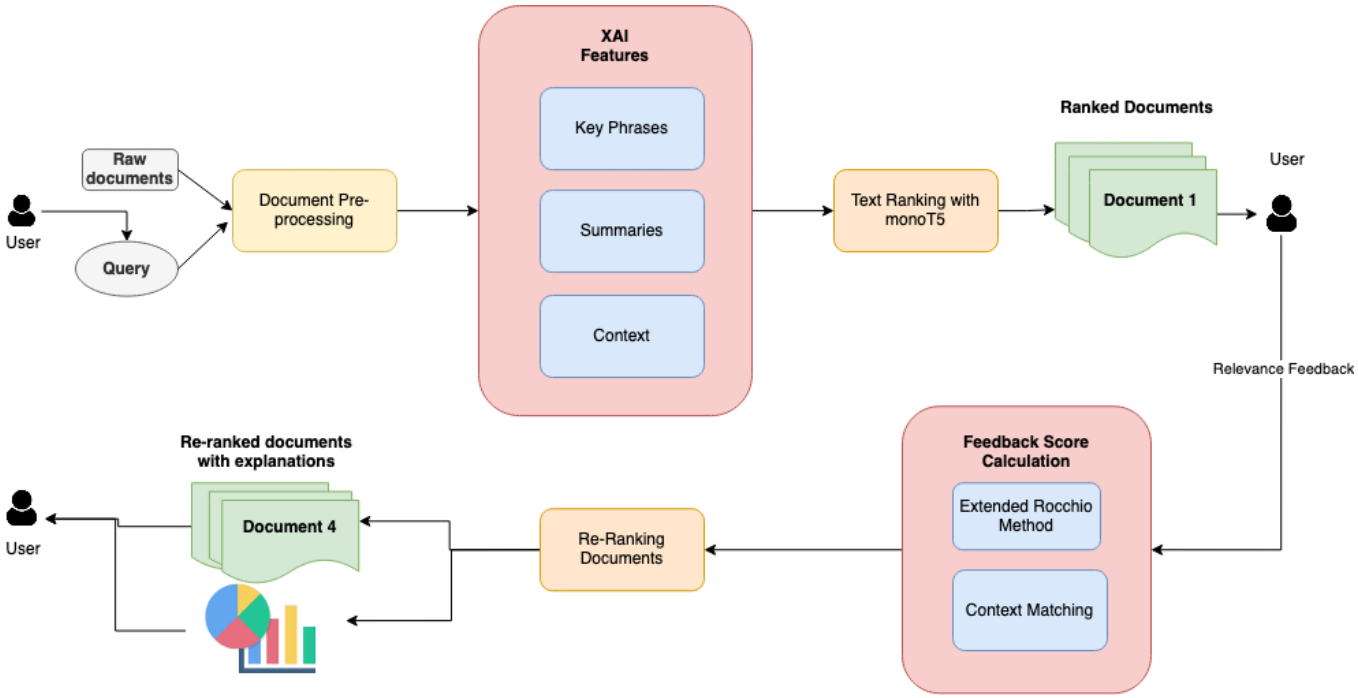

Fig. 5. The architecture used for our experiments. The user starts the search by querying in the front end as shown from the left. The direction of the arrows shows the direction of the flow of data.

Text: *Recent evidence suggests that critically ill patients are able to tolerate lower levels of haemoglobin than was previously believed. It is our goal to show that transfusing to a level of 100 g/l does not improve mortality and other clinically important outcomes in a critical care setting. Although many questions remain, many laboratory and clinical studies, including a recent randomized controlled trial (RCT), have established that transfusing to normal haemoglobin concentrations does not improve organ failure and mortality in the critically ill patient. In addition, a restrictive transfusion strategy will reduce exposure to allogeneic transfusions, result in more efficient use of red blood cells (RBCs), save blood overall, and decrease health care costs.*

Summary: Recent evidence suggests that critically ill patients are able to tolerate lower levels of haemoglobin than was previously believed. Although many questions remain, many laboratory and clinical studies, including a recent randomized controlled trial (RCT), have established that transfusing to normal haemoglobin concentrations does not improve organ failure and mortality in the critically ill patient.

Fig. 6. *Summarization of a text.*

GitHub[3] is shown in fig:7.

Text: *Recent evidence suggests that critically ill patients are able to tolerate lower levels of haemoglobin than was previously believed. It is our goal to show that transfusing to a level of 100 g/l does not improve mortality and other clinically important outcomes in a critical care setting. Although many questions remain, many laboratory and clinical studies, including a recent randomized controlled trial (RCT), have established that transfusing to normal haemoglobin concentrations does not improve organ failure and mortality in the critically ill patient. In addition, a restrictive transfusion strategy will reduce exposure to allogeneic transfusions, result in more efficient use of red blood cells (RBCs), save blood overall, and decrease health care costs.*

Key-phrases: transfusing, haemoglobin, ill patients, mortality, red blood cells

Fig. 7. *Key-phrasification of a text.*

Context Extraction: Post key-phrasification, we extract context from 'abstract' for each key-phrase and store them in a separate field. By context, we mean a window that comprises 5 words before and after the key-phrase locate in the text. Note that some of the keywords might not be present in the text. We do not extract any context for such key-phrases. We later use these contexts in the re-ranking with relevance feedback phase. Refer fig:8 for an example of contexts extracted from a text with given key-phrases.

Context-1: "*our goal to show that **transfusing** to a level of 100) , have established tha **transfusing** to normal haemoglobin concentrations does*"
Context-2: "*to tolerate lower levels of **haemoglobin** than was previously believed . established that transfusing to normal **haemoglobin** concentrations does not improve organ*"

Fig. 8. *An example of contexts extracted for the text and key-phrases shown in fig:7*

Since key-phrasification, context extraction, and summarization takes a lot of time to process, we extract and store them to run the process smoothly and reduce the delay in the system.

C. Text Ranking

At first, we tested learning to rank state-of-the-art-text-to-text-transformers, MonoT5[13] and DuoT5[14], against the base model BM25. For our use case, we used PyTerrier Python transformers plugins [4] with our pre-processed TREC-COVID-19 dataset since we had the relevance judgments for some pre-defined queries. We performed experiments by building the PyTerrier pipeline [5] and running PyTerrier experiments

[3]https://github.com/keyphrasification/hands-on-with-pke

[4]https://github.com/terrierteam/pyterrier_t5

[5]https://pyterrier.readthedocs.io/en/latest/pipeline_examples.html

API [6]. We used different metrics to evaluate the models and selected the best-performing model on this particular dataset.

- **Mean Average Precision (MAP):** MAP is an evaluation metric in Information retrieval that is used to measure the average precision score for a query set with ground truth. It computes the average precision for each query in the set and then calculates the mean over all queries. The value of MAP lies from 0 to 1 and higher scores indicate better performance.
- **Reciprocal Rank (Recip Rank):** Reciprocal rank is a measure of the highest-ranked relevant document for each query. It calculates the inverse rank of the first relevant document and takes the mean over all queries. Reciprocal rank ranges from 0 to 1, with higher scores indicating better performance.
- **Normalized Discounted Cumulative Gain at 10 (NDCG Cut 10):** It is a measure of the quality of the ranked list of retrieved documents. It calculates the relevance of each document at each rank position and applies a discount factor based on the position. It takes the mean over all queries and normalizes the result by the ideal DCG cut 10 score, which is the score achieved by a perfect ranking of the relevant documents. NDCG cut 10 ranges from 0 to 1, with higher scores indicating better performance.

MonoT5 outperformed BM25 and DouT5 in all the metrics refer the Table I for experiment results. It has a smaller model size than DuoT5 and is faster to train and infer. MonoT5 can be fine-tuned on a ranking task using a triplet loss function or another suitable loss function to learn to rank text based on the query. The model estimates a score s_i quantifying how relevant a candidate $d_i \epsilon R_{i-1}$ is to a query q. That is:

$$P(Relevant = 1/d_{i,q})$$

Also, note that sequences longer than the model's maximum of 512 tokens are truncated. Due to limited resources and faster response demand in the case of humans in the loop, we found MonoT5 to be a better choice for re-ranking text. For a detailed description of MonoT5 one can refer Nogueira et al., 2021 [14].

TABLE I
EXPERIMENTS WITH DIFFERENT IR MODELS ON TREC-COVID-19 DATASET

Model Name	MAP	Recip rank	P10	NDCG CUT 10
BM25	0.076117	0.807094	0.678	0.601847
MonoT5	0.081632	0.869000	0.740	0.676241
DuoT5	0.015044	0.813000	0.720	0.646023

After selecting MonoT5 as our search model, we configured it to access the indexed TREC-COVID-19 data for ad hoc search. The user input query is used to rank documents using MonoT5. Here, we search for only the top 50 results and then we send these results to the front-end through Python Flask API.

[6]https://pyterrier.readthedocs.io/en/latest/experiments.html

D. Collecting Feedback

We collect feedback from users in the front-end after the text ranking stage which shows top 10 results only. The user sees the result with the title, key-words, also known as key-phrases, highlighted in the summary. The user can also access the full 'abstract' text where key-phrases are also highlighted. Since MonoT5 is a point-wise re-ranker that uses query for relevance classification, we also highlighted the query words in the summary as well as the full text. All the non-selected documents are among the top 10 as irrelevant and the ones marked as relevant. The rest 40 documents are treated as none, which means neither as relevant nor as irrelevant. The response is sent to the back end with appropriate flags.

E. Re-ranking with Relevance Feedback

We process the feedback response from the user and divide the 50 records into 3 separate categories - relevant (marked by the user in top 10), irrelevant(marked by the user in the top 10), and neither (the rest 40 documents). Now we use the Rocchio algorithm to enhance the user's query which includes the relevant and irrelevant documents as part of the calculation. Rocchio algorithm was developed using a vector space model hence we converted the query, 'abstract' texts, and key-phrases into vectors. We took the following steps after we receive the response from the user -

1. Vectorization: For vectorization, we used the Gensim Doc2Vec model which is an open-source Python library. Doc2Vec is a model that simply represents each document as a vector. You need to set the hyper-parameters like the dimension of the vector. We trained the model using the text present in the 'abstract' field of the dataset. Prior to the training, we saved the model in pickle format. This model can be used to generate vector embedding of any given text document. While the system is running, we obtain the vectors of document text and also the key-phrase list which we supply to the model as a string.

2. Extended Rocchio Algorithm: We extended the concept of Rocchio algorithm and added more terms in the equation 1. To enhance the query with the information from key-phrases, we added a weighted sum of vector key-phrases of relevant and non-relevant documents. Hence the equation in 1 becomes -

$$\vec{Q}_m = \alpha\vec{Q}_0 + \frac{\beta}{|D_r|}\sum_{\vec{D}_j \in D_r}\vec{D}_j - \frac{\gamma}{|D_{nr}|}\sum_{\vec{D}_k \in D_{nr}}\vec{D}_k + \frac{\delta}{|P_r|}\sum_{\vec{P}_j \in K_r}\vec{P}_j - \frac{\eta}{|P_{nr}|}\sum_{\vec{P}_k \in P_{nr}}\vec{P}_k \quad (2)$$

where P_r and P_{nr} are sets of vectors of relevant and non-relevant key-phrases respectively. Whereas, the hyper-parameters δ and η are synonymous to the hyper-parameters β and γ.

3. Feedback score calculation: The enhanced query vector is then used to calculate cosine similarity between the 40 un-marked documents and the relevant documents selected by the user. We also store a list of key-phrases selected by the user to count the occurrences of the key-phrases in the context of each document. Then, we take the aggregate of all the occurrences of key-phrases and store it as the key-phrase context match count. Finally, the feedback score of each document is calculated as the linear combination of the normalized cosine similarity and normalized key-phrase context match count.

$$feedbackScore = a \cdot \frac{cosineSimilarity}{\parallel cosineSimilarity \parallel} + b \cdot \frac{key-phraseContextCount}{\parallel key-phraseContextCount \parallel}$$

we determined the hyper-parameters a and b by doing a number of experiments.

At last, the documents are sorted in the increasing order of their feedback score and are returned to the user as a result of re-ranking with relevance feedback.

* *Feedback-score hyperparameters selection:* These percentages *a, b* behave as hyperparameters in terms and determine the relative importance of the cosine similarity and key-phrase similarity in the re-rank task. The values of a and b might be in the range of 0 to 1, and choosing the right numbers is essential for obtaining optimal outcomes. Through trial and error over a range of values and query terms, we tried to optimize these hyperparameters. As seen in Fig:10, we reach the conclusion of keeping their values as 30% and 70% respectively as at this value every query shows at-least one user-selected relevant document in the re-rank result.

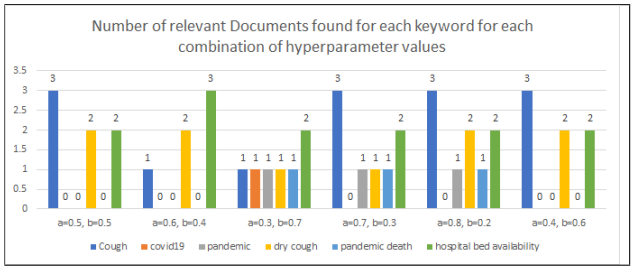

Fig. 9. Tuning Feedback Score hyper-parameter values

F. Generating explanations

Now, to provide an explanation of the system's rationale behind the re-rank approach at the individual instance level we employed a bunch of strategies.

Global level:

1) Graphical representation of Query vector modification: To explain the ranking to the user, we generated a graphical representation of vectors used in the calculation of the extended Rocchio algorithm. The graph in Fig:4 shows a projection of relevant and irrelevant documents, and query and modified query. The scatter plot is intended to make the user understand how the modified query vector is now closer to the relevant documents which we received from the user's relevance feedback. The graph shows the mathematical outcome of Rocchio algorithm.

Local level:

1) Highlighted Query: The queries built by key-terms are simple expressions of the user's information need and do not contain any complex grammatical conjunctions. Subsequently, the terms are considered to be in a logical 'AND' relationship and contribute equally to the determination of resulting documents. As an example in Fig:1, we searched for 'pandemic death' and highlight the query-terms in bold. The user may aim towards those documents containing this query as relevant and those that do not contain it as irrelevant.
2) Highlighted Key-Phrases: Through a text analysis model, we extract the most relevant word and expressions used in a document.
3) Highlighted the context-match: Representing how many times any key-phrase of the selected documents appears in the vicinity of 5 word-context of the key-phrases of every other document. The context-match is highlighted in the re-rank list as well for the user to identify them easily.
4) Bar chart for context-match count: The context-match count is represented through a bar-chart that explains how the key-phrase word match in context count has contributed to feedback score calculation.

V. Working with RelEx

RelEx is a self-explanatory application that administers its users to perform certain tasks at every step of its progression. We categorize the tasks in certain levels for ease of assessment for the users.

A. Pre-feedback stage:

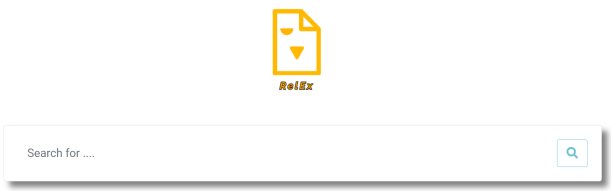

Fig. 10. A screenshot of Search bar

At this initiation level, the user is presented with a clean slate view with a search bar and asked to enter their information need, which then takes them to the search result page. At

this stage, the user is asked to assess each document carefully and with the help of highlighted query and key-phrase alongside detailed document content. For ease of assessment, our application only displays the top 10 documents to choose from. Following assessment, the user checks the 'Relevant' boxes to select their 'chosen relevant documents'. There is a possibility for the user to not find any pertinent document, in that case, the user can opt to select the 'Select all as irrelevant' radio button. Similarly the 'Select all as relevant' can also be chosen in case all the documents are relevant. In these cases, there is no scope left for the system to perform any further exploratory and explanatory operation, hence the user is prompted to enter a different query instead. As soon as the user submits their selections our re-rank process gets initiated.

B. Post-feedback stage:

This stage kicks off by presenting the revised search result to the user as shown in Fig:1. The cue here (Fig:11) directs its users towards the explainable elements that would help interpret the re-rank process. Following this user can explore the elements as shown in Fig:3,4.

Fig. 11. A screenshot of task prompt in feedback result. Users can expand each of the elements and get a cue about the explainable elements

VI. EVALUATION

We conducted a user study to determine how the levels of explainability can be measured for different groups of users. Our evaluation focuses on the quality of produced interpretation in a more general setting and gauges how well the general notions are captured.

A. Approach

We explained the aim of our user study through a brief introduction to the participants followed by instructions to fulfill two agendas in parallel -

- Users were asked to carry out exploratory search tasks in an experimental system. Our experiment is based on the TREC-COVID-19 (CORD-19) dataset hence we specifically requested the user to use COVID-19 related search queries.
- They were provided with a Google form containing a questionnaire regarding the outcomes and judgments and they were instructed to evaluate the outcomes in the form accordingly. For multiple quantitative questions, they were asked to rate on a 7-point Likert scale starting from 1 (Strongly Disagree/Least helpful) to 7 (Strongly Agree/Extremely helpful). Similarly short textual explanations were asked for some qualitative questions to make an assessment of the responses later.

We observed user search activity and gathered evaluations of search results. With the goal to make our evaluation comparable, we followed the approach taken by Mi and Jiang et al., 2019 [1] and Chios and Verberne et al., 2021[8], we chose that technique.

- **[Explainability]** - "Looking at the explanations given on the final resultant page, I can understand the ranking of the results for the given query based on my feedback."
- **[Assessability]** - "By looking at the feedback page, I can tell which results are relevant to my query without opening the documents."

B. Design

In order to enable users to get a good conceptual grasp of our application, we developed two comparable interfaces. One with explanation features and the other without. The dual interface method intends to give users a broader understanding of the benefits and drawbacks of each interface, making it easier for them to assess and choose the best choice based on their unique needs and preferences.

- **System 1 (Regular Interface without explainable features)** Fig:13 resembles the simplest and most basic search engine interface with its document title and summary.
- **System 2 (Explainable Interface)** Fig:14 is our experimental resultant interface which contains 3 additional elements to support the explanations besides the document title and summary. The Query terms are highlighted in a different background and the individual document key phrases are marked with bold faces.

We invited 30 people to participate in our unpaid user study and 13 consented to take part. We organized an online single-session user study for each user and instructed them to perform two distinctive tasks with clearly defined goals at the end. We have 3 main stages for users to explore in both systems.

1) *Search/Home Page and evaluation:* At the introductory stage, we collected some basic information about the user e.g., their background, and the extent to which search engines are integrated into the user's daily life. In order to address the explicit relevance feedback method we also specifically ask the users about their knowledge regarding COVID-19. We briefly introduce our experiment and evaluation expectations on this page. To increase task diversity we encouraged the users to try out different queries related to Covid-19 for each interface multiple times.
2) *Feedback Page:* The top 10 results of the user query are shown on this page for both of the systems, We want to evaluate the "assessability" of both systems by allowing users to read and judge the abstract and if needed the content as well. The users are needed to perform a comparative analysis of both systems and

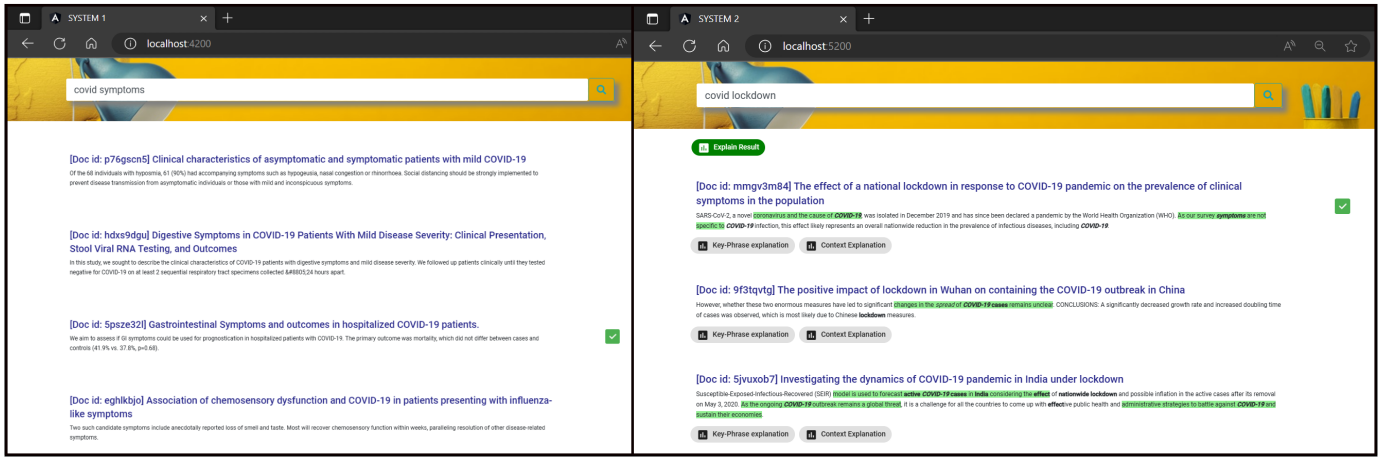

Fig. 12. **Left:** A screenshot of the resultant page of System 1 interface with no explainability. **Right:** A screenshot of the resultant page of the System 2 interface with explainability.

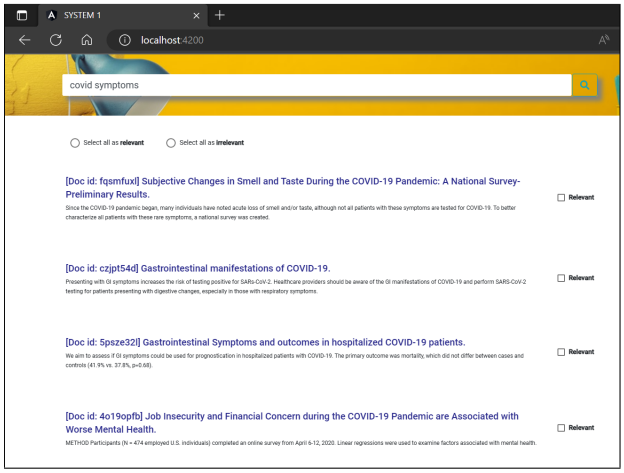

Fig. 13. A screenshot of Feedback page of System 1 interface.

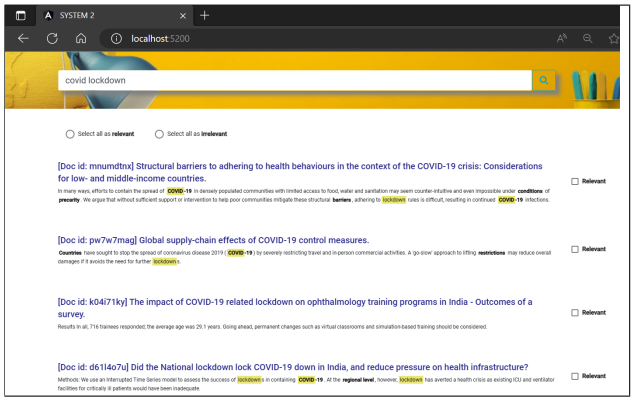

Fig. 14. A screenshot of Feedback page of System 2 interface.

reflect on the efficacy of the key-phrase and summary provided in System 2.

3) *Resultant page judgements:* After processing the relevance feedback, we display the re-ranked top-10 documents in decreasing order of the feedback score. At this stage, we asked the user to make sense of the reasoning behind the order of the documents. In System 1 there is no explanation provided Whereas, in System 2, we display the explainable elements. The users are asked to rate the explainability of the re-ranked results according to their understanding.

C. *Results and discussion*

In this section, we will discuss the results of our user study and draw conclusions to the research questions.

RQ1. How well do the key-phrases in comparison to the summary of a document help user select relevant documents during Relevance Feedback?

In order to answer this question, we asked users to rate if the *highlighted key-phrases* of each document helped them in understanding the relevance of their query in the feedback page of System 2. The users, having a choice to read the full text or take help from the highlighted key-phrases, are needed to rate how much it helped them in selecting the current document as relevant on a Likert scale (0-7).

- Users gave mean rating of *6.25 out of 7 (standard deviation = 0.97)* for the *highlighted key-phrases*.
- Users also rated the *summary* with a mean of 6.08 out of 7 (with a standard deviation of 0.79).
- Users, who did not open the document in the feedback page in System 1, rated the *summary* with a mean of 5.5 out of 7 (with a standard deviation of 0.79) being helpful.
- Users, who did not open the document in the feedback page of System 2, rated the *summary* with a mean of 5.5 out of 7 and gave the same mean rating for *highlighted key-phrases*. Whereas, Users who opened the document to read the full text in *abstract* found it very helpful and gave a mean rating of 6.71 out of 7 for the *highlighted key-phrases*.

Figure: 15 illustrates the results obtained from the assessability of selecting the relevant document of *key-phrases* and

summary in System 2. However, when we performed WSR (Wilcoxon Signed-Rank) test on two sets of ratings (*summary* and *key-phrases*) for System 2 to test the null hypothesis that the median difference between the paired values is zero, we obtained a test statistic of 13.5 and the p-value of 0.4795. In this case, since the p-value is greater than the conventional significance level of 0.05, we do not have sufficient evidence to reject the null hypothesis. This means that there is no significant difference between the two sets of ratings. Thus, based on the Wilcoxon Signed-Rank test, we cannot conclude that one set is more helpful than the other in terms of assessability.

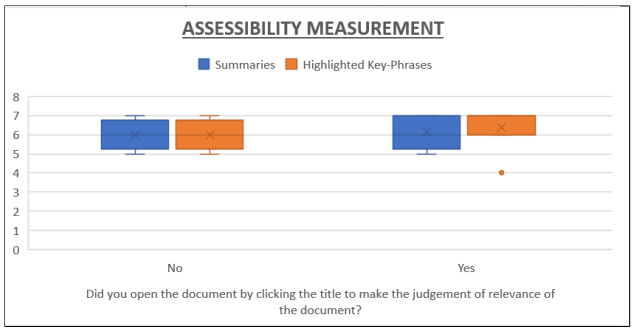

Fig. 15. Assessability in System 2 interface.

Hence, we can conclude that the users have considered the highlighted key-phrases to be more helpful in document selections compared to summaries.

RQ2. How do users rate the explanations in our explainable interface in comparison to a standard interface, after Relevance Feedback?

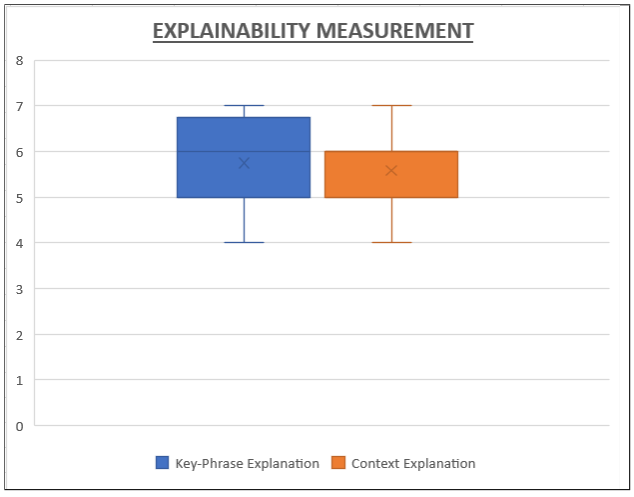

Fig. 16. Explainability of Key-phases and context matching in System 2 interface.

We answered this research question by providing elements of explanation in the final resultant page. We asked participants to rate the helpfulness of explainable components in explaining the ranking of the documents as per their feedback on the Likert scale of 0 to 7.

- Users rated *Key-phrase explanation* with a mean rating of 5.75 out of 7 (with a standard deviation = 0.97).
- And *context explanations* were rated with a mean rating of 5.58 out of 7 (with a standard deviation of 0.79).
- For System 1 (Regular Interface), 25% of participants voted that they understand the ranking principle, and 75% voted that they cannot understand why certain documents are ranked higher than the others.
- Our *Explain Result* proved to be non-intuitive to the participants and failed to give a global explanation.
  We have illustrated the dispersion graph of users' ratings in figure 16.

On the basis of the results of the evaluation we deduce that the graphical representation of Rocchio algorithm does not meet our expectations and failed to explain the re-rank logic. On the other hand, key-phrase and context explanations helped the users in comprehending the rank principle.

RQ3. Does the extended Rocchio Algorithm provide the user with more pertinent results than the original Rocchio Algorithm?

Followed by the online user study we also conducted an experiment to determine whether the expanded algorithm was more effective than the conventional method in producing results that were more pertinent and with better precision. During our user study, we asked the user to capture the query and the document Ids of selected relevant documents. Later we replicated those query searches in two systems. In the first interface, we used the additional Rocchio algorithm 1, and in the second interface utilizes the extended version 2. We can see from Fig: 17 that interface A has consistently outperformed interface B in terms of retrieving more selected relevant documents. For example, for the query 'Long covid symptoms', in interface A 2 out of 3 selected documents were present in the re-ranked feedback page, whereas none were present in the case of interface B.

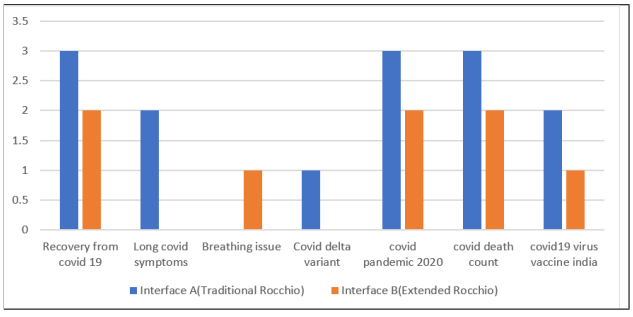

Fig. 17. Comparison of the results returned by traditional Rocchio algorithm vs. extended Rocchio algorithm for the same query.

Based on the evaluation of this experiment we could determine that the extended Rocchio outperforms the traditional Rocchio algorithm in terms of producing more numbers of pertinent data.

VII. Limitations of current work & future scope

We dealt with challenges that occasionally presented themselves as a barrier to the deployment of functionalities as we advanced through the development and testing phase. Moreover, we received critical reviews during our user evaluation as well which helped us decide our future scope of research. Following are a few significant roadblocks and scopes of future works that we encountered at some stages -

- **Optimization constraint:** RelEx is capable of successfully completing the specified task, but it might not perform at its best in terms of time taken result recovery. It takes around 50 to 60 secs to generate the first set of rankings and approximately another 30 seconds to generate the re-ranked result. Firstly, it was difficult to thoroughly optimize the complicated underlying algorithms and data structures due to time restrictions and the unavailability of computational resources throughout the project development period. We expect to optimize our system's responses in the future to provide faster results.
- **Flask application port conflicts:** Having to change the port number of the Flask application every time we made code changes was a significant issue in this project. Although it seemed like a trivial task we quickly realized that it was consuming a significant amount of time during our continuous integration process and was the reason for a lot of loss of effort and time during the process.
- **Query constraints:** The vocabulary related to Covid-19 is constantly evolving and enlarging. Hence, the inclusion of all possible query terms and spell-check proved to be a difficult task due to the enormous number of possible spelling variations. Our system only accepted correctly spelled, grammatically accurate queries. Employing techniques such as phonetic matching and approximate string matching, fuzzy matching algorithms were not supported in a small-scale project like ours.
- **Rocchio Classification plot:** Rocchio algorithm when visualized through PCA plot should ideally show clusters of relevant and irrelevant documents and the modified query vector will appear as being nearer the centroid of relevant documents. Although our evaluation result clearly shows that the plot does not accurately depict how documents are divided and grouped into distinct categories as expected and the modified vectors are not always closer to the relevant vector cluster either.
- **All relevant or all irrelevant documents:** In case the user finds all the resulting documents as relevant or irrelevant there is not any scope of improvement left for the system. Hence our application does not proceed further toward the re-rank stage, instead prompts the user to try a new query. We intend to modify this functionality in two ways. Either the query gets tweaked and a separate set of results is shown that the user might find relevant or the system subsequently displays the next 10 documents. And at last a greater number of participants might also make it possible to apply statistical testing of hypotheses to compare systems when deciding on the final results.

VIII. Conclusion

With an aim to mitigate the limitations of traditional search engines, we proposed an XAI search engine RelEx with Relevance Feedback for exploratory search. We came up with an extended version of the Rocchio algorithm and provided empirical evidence that the rankings adapted well to the user's feedback for an ad-hoc query. And, more number of the selected relevant documents were produced on the resultant page compared to the traditional Rocchio algorithm. We introduced key-phrases and context matching as explainable elements in RelEx and compared the results with the regular search engine. The explainability of RelEx has proven to be better for users based on their subjective experiences. It is important to explicitly indicate when documents are considered relevant based on matches to the user's personal preferences of key-phrases. In the feedback page UI, users relied on key-phrases as well as *summaries* to decide whether the documents are relevant or not and find more helpful in the *abstract* where they did not feel the need to read the whole text. There was no significant difference between the ratings of *key-phrases* and *summary* in terms of assessability. Overall, RelEx has proven to have significant utility in the field of explainable search engines. However, we have identified potential areas of modification in order to unleash the full potential of our application. A few crucial scopes of future work are performance optimization and the exploration of advanced classification algorithms.

References

1 Mi, S. and Jiang, J., "Understanding the interpretability of search result summaries." Association for Computing Machinery, Inc, 7 2019, pp. 989–992.

2 Casalegno, F., "Learning to rank: A complete guide to ranking using machine learning," (Accessed on 05/01/2023). [Online]. Available: https://shorturl.at/lsIRX

3 Schutze, H., Manning, C. D., and Raghavan, P., *Introduction to information retrieval*. Cambridge University Press, 2008.

4 "How do you evaluate the effectiveness of relevance feedback in information retrieval?" https://www.linkedin.com/advice/0/how-do-you-evaluate-effectiveness-relevance-feedback-information, (Accessed on 05/01/2023).

5 White, R. W., Jose, J. M., and Ruthven, I., "Comparing explicit and implicit feedback techniques for web retrieval: Trec-10 interactive track report."

6 Rocchio Jr, J. J., "Relevance feedback in information retrieval," *The SMART retrieval system: experiments in automatic document processing*, 1971.

7 "Rocchio algorithm - wikipedia," https://en.wikipedia.org/wiki/Rocchio_algorithm, (Accessed on 05/01/2023).

8 Chios, I. and Verberne, S., "Helping results assessment by adding explainable elements to the deep relevance matching model," 6 2021. [Online]. Available: http://arxiv.org/abs/2106.05147

9 Polley, S., Janki, A., Thiel, M., Hoebel-Mueller, J., and Nuernberger, A., "Exdocs: Evidence based explainable document search," 2021. [Online]. Available: http://ceur-ws.org

10 Voorhees, E., Alam, T., Bedrick, S., Demner-Fushman, D., Hersh, W. R., Lo, K., Roberts, K., Soboroff, I., and Wang, L. L., "Trec-covid: constructing a pandemic information retrieval test collection," in *ACM SIGIR Forum*, vol. 54, no. 1. ACM New York, NY, USA, 2021, pp. 1–12.

11 Wang, L. L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., Funk, K., Kinney, R. M., Liu, Z., Merrill, W., Mooney, P., Murdick, D., Rishi, D., Sheehan, J., Shen, Z., Stilson, B., Wade, A., Wang, K., Wilhelm, C., Xie, B., Raymond, D., Weld, D. S., Etzioni, O., and Kohlmeier, S., "Cord-19: The covid-19 open research dataset," *ArXiv*, 2020.
12 Savary, A. *et al.*, "Proceedings of the 58th annual meeting of the association for computational linguistics (acl 2020): Tutorial abstracts," 2020.
13 Nogueira, R., Jiang, Z., and Lin, J., "Document ranking with a pretrained sequence-to-sequence model," *arXiv preprint arXiv:2003.06713*, 2020.
14 Pradeep, R., Nogueira, R., and Lin, J., "The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models," *arXiv preprint arXiv:2101.05667*, 2021.