
Indian Statistical Institute

Master's Thesis

---

# Unified Framework for Pointwise Explainable Information Retrieval

---

*Author:*
Harsh Agarwal

*Supervisor:*
Dr. Debapriyo Majumdar

*A thesis submitted in fulfillment of the requirements for the degree of Master of Technology*

*in*

Computer Science

June 16, 2024

**Indian Statistical Institute**
203, B.T. Road, Kolkata : 700108

# CERTIFICATE

I certify that I have read the thesis titled **Unified Framework for Pointwise Explainable Information Retrieval**, prepared under my guidance by **Harsh Agarwal**, and in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of **Master of Technology in Computer Science** of the Indian Statistical Institute.

**Debapriyo Majumdar**
Assistant Professor
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute

Kolkata
June, 2024.

# Declaration of Authorship

I, Harsh Agarwal, declare that this thesis titled, "Unified Framework for Point-wise Explainable Information Retrieval" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Master's degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Harsh Agarwal

Date: 16/06/2024

INDIAN STATISTICAL INSTITUTE

# *Abstract*

Computer Science

Master of Technology

**Unified Framework for Pointwise Explainable Information Retrieval**

by Harsh Agarwal

As Machine Learning (ML) models become increasingly sophisticated and opaque, the necessity for explainability to ensure transparency and accountability in their applications grows. Despite numerous proposed methods for explaining these complex models, there remains a lack of a unified framework that encompasses these approaches for comprehensive experimentation and analysis. This thesis introduces "`ir_explain`", an integrated Python module that consolidates various explainability techniques specifically for Information Retrieval (IR). While the entire module represents a collaborative effort, my focus has been on the implementation and analysis of Pointwise explanations. By consolidating these methods into a single package, `ir_explain` simplifies their application and facilitates robust analysis. Through a series of experiments, this thesis showcases the module's practicality and effectiveness, contributing to the development of more transparent and interpretable ML models in the IR domain, with a primary focus on Pointwise explanations.

# *Acknowledgements*

I would like to express my sincere gratitude to my research guides, Dr. Mandar Mitra and Dr. Debapriyo Majumdar, for their invaluable guidance and support throughout this research. Their expertise and insights have been instrumental in shaping this thesis.

Special thanks to Sourav Saha, research scholar pursuing his Ph.D, who guided me through various aspects of this research. His patience and knowledge were crucial in overcoming numerous challenges.

I would also like to thank my classmate and colleague, Swastik Mohanty, for his collaboration. His support and cooperation have made this journey more manageable and enjoyable.

Thank you all for your patient support and belief in my work.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **XAI** | e**X**plainable **A**rtificial **I**ntelligence |
| **IR** | **I**nformation **R**etrieval |
| **ExIR** | **Ex**plainable **I**nformation **R**etrieval |
| **LIME** | **L**ocal **I**nterpretable **M**odel-agnostic **E**xplanations |
| **EXS** | **EX**plainable **S**earch |
| **LIRME** | **L**ocally **I**nterpretable **R**anking **M**odel **E**xplanation |

# Chapter 1

# Introduction

Machine Learning (ML) models have become exceedingly advanced and complex, often reaching scales of billions of parameters, that render them incomprehensible to human understanding. These sophisticated models are typically perceived as "black boxes" where the internal decision-making processes are opaque and difficult to interpret. Despite this, such models are increasingly deployed in critical applications ranging from healthcare to finance, necessitating a higher level of transparency, regulation, and accountability.

With these concerns in mind, this thesis focuses on one specific domain: Information Retrieval (IR). We begin by briefly introducing IR and formally defining key Explainable IR (ExIR) concepts in this chapter. In Chapter 2, we delve into Pointwise Explanations in greater detail. Chapter 3 presents the `ir_explain` library, which we developed to integrate various explainable IR techniques into a single Python module. In Chapter 4, we demonstrate the ease of use of this module and conduct experiments using Pointwise explainability techniques. Finally, Chapter 5 concludes the thesis by summarizing the key points discussed and offering suggestions for future research directions.

## 1.1 Information Retrieval

Information Retrieval (IR) is a field of computer science focused on obtaining relevant information from large collections of data, such as databases, the internet, or digital libraries. It involves developing algorithms and systems that search, filter, and rank information based on user queries, ensuring efficient access to pertinent data. Information Retrieval permeates every part of our lives today by enabling seamless access to relevant information across various digital platforms, from search engines and social media to streaming services and online shopping. For example, platforms like Spotify and YouTube leverage

IR [9, 11, 3] to enhance user experience through efficient search functionalities and personalized content recommendations. By analyzing user behavior, preferences, and query patterns, these platforms deliver tailored playlists and video suggestions, improving user satisfaction, engagement, and content discovery.

## 1.2 Explainable Information Retrieval

Given how ubiquitous IR has become, it becomes important to take a step back and understand how the underlying models work. Is the data used to train the models fair and unbiased? Are the incentives of the people providing the aforementioned services aligned with those who are using them?

To understand the importance of the former, let us think of job searching platforms, where CVs or resumes are often deemed relevant or "fit" for a particular job using IR techniques. It's necessary to know that the models make such a decision based on skills and metrics relevant for the job instead of some underlying social or economic bias [10].

Commercial incentives can significantly affect Information Retrieval (IR) by prioritizing content that maximizes user engagement and click-through rates, often at the expense of relevance and quality. Algorithms may be tuned to favor ads or sponsored content, leading to a biased ranking of search results and recommendations. This can result in users being presented with commercially driven content rather than purely relevant information, potentially diminishing the user experience.

This forms a strong basis for the need for Explainable Information Retrieval. When other incentives influence IR systems, it becomes crucial to have transparency in how search results and recommendations are generated. Explainable IR can help users understand why certain content is prioritized, ensuring that the system's decision-making process is clear and trustworthy. This transparency can mitigate potential biases introduced by commercial interests.

Explainable Information Retrieval also aids model developers by revealing inherent biases within algorithms and datasets. Through transparency in decision-making processes, developers can identify and comprehend biases present in data or introduced during training. This insight enables deliberate actions to be taken to address biases, thereby promoting fairness and equality.

## 1.3 Formalizing Explainable AI concepts

**Interpretability** can be defined as "the ability to explain or to present in understandable terms to a human" [4]. In literature, interpretability and **explainability** are often used interchangeably. Thus, Explainable IR is the attempt to make the workings of complex black box models used in modern IR more understandable to a human.

**Inherent interpretability:** Certain IR models are naturally understandable to humans. For example, it is easy for humans to understand how Decision Trees generate outputs. The decision made for splitting at each internal node can be tracked down to arrive at the final output. Sparse linear models, such as Linear Regression are also considered inherently interpretable, since they model linear relationships between input features and the output which is relatively simple for a human to understand.

**Global vs. Local interpretability**: Global interpretability refers to the ability to understand the overall behavior or decision of an ML model. It aims to provide insights into how the model makes decisions across all possible (or most) inputs. Local interpretability focuses on understanding the model's behavior for individual predictions or small regions of the input space. It provides explanations for *specific* instances, explaining why the model made a particular decision for a given input.

**Model-agnostic explanations:** Model-agnostic explanations provide insights into the decision-making processes of machine learning models without requiring access to their internal workings. They treat the model to be explained as a black box, without making any assumptions about the model architecture or parameters. These type of explanations are extremely useful when one does not have access to the model itself, but only to the outputs produced by the model. This is true in many real life applications, such as search engines, where the users do not have access to the ranking model but only to the ranked results provided by the search engine.

**Post-hoc explanations:** They primarily seek to explain the output generated by an ML model, rather than explain the inner workings of the model. Post-hoc explanations work by analyzing the relationship between the input features and the output of the model after the model has been trained. They do not require access to the internal architecture or the training process of the model.

## 1.4 Standard paradigm in modern neural IR

In the general setting of Information Retrieval, users type in questions or keywords, and the system searches through all available documents to find the ones that best match the user's query. The standard approach in modern Neural Information Retrieval employs a two-stage process. Initially, a sparse retrieval model like BM25 is used as the first stage ($M_1$) to retrieve the top $k$ documents based on the given query $Q$. The value of $k$ is selected to be significantly smaller than the total number of documents in the collection. In the second stage ($M_2$), dense retrieval methods are then applied to re-rank this subset of $k$ documents. The first stage of the pipeline is considered to be *inherently intrepretable* in literature, as discussed. However, the second stage re-ranking by complex neural models are often inscrutable.

Based on this paradigm, post-hoc ExIR attempts to explain the re-rankings obtained by the dense retrieval models. Furthermore, post-hoc Explainable IR can be broadly grouped into the following three categories, i) pointwise explanations, ii) pairwise explanations, and iii) listwise explanations. Figure 1.1 shows these three explantion categories.

### 1.4.1 Pointwise explanations

Pointwise explanation considers a particular document and aims to explain the document's score (in terms of relevance to the query) or position in the list of documents retrieved by the complex retrieval model, for the given query. In this framework, each document is considered independently. Given a query $Q$, let us consider that a document $D$ was given a score $D_s$ and retrieved at a rank $D_r$ by the complex model $C$. The pointwise explanation aim to explain why the particular document $D$ was retrieved at a rank $r$.

### 1.4.2 Pairwise explanations

Pairwise explanation considers a pair of documents and aims to explain why one of the documents was given a higher score (in terms of relevance to the query) or a higher position in the list of documents retrieved by the complex retrieval model, than the other, for the given query.

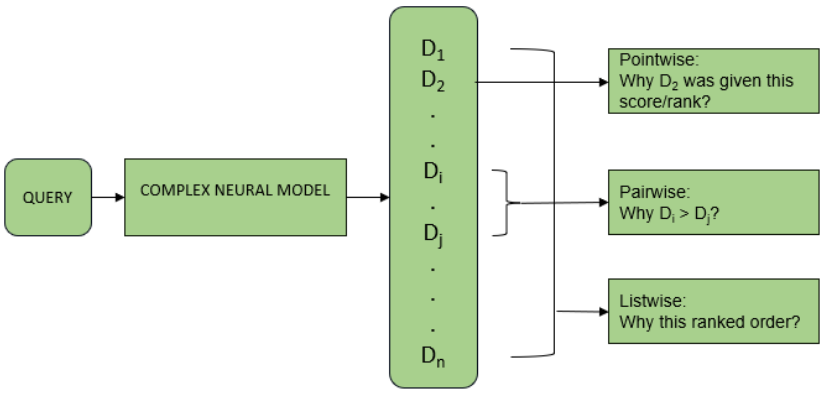



FIGURE 1.1: Three approaches of ExIR, namely, pointwise, pairwise and listwise explanations.

### 1.4.3 Listwise explanations

Listwise Explanations consider the entire ranked list of documents retrieved by the complex ranking model $C$. Given a query $Q$, we aim to explain the ranked list of top $k$ documents retrieved by $C$.

# Chapter 2

# Pointwise Explanations

## 2.1 Problem setting

Given a query $Q$, a document $D$, and a complex neural reranking model $C$, our goal is to comprehend why the reranking model assigns a specific score to this document. Alternatively, given a list of ranked documents, we seek to understand why the reranking model assigns a particular rank to this document.

## 2.2 Baseline methods

In the survey for ExIR [13, 2], have shown several baseline methods for Pointwise Explanations. We focus here on two of these popular methodologies.

### 2.2.1 Local interpretable model-agnostic explanations

Local model-agnostic interpretability techniques generate explanations for individual decisions by approximating the local behavior of a complex black-box model with a simple linear model [12]. This is achieved by training the linear model in an interpretable feature space using training data generated through perturbations of the instance to be explained, constrained by locality, and querying the black-box model for labels.

Consider a complex classifier $C$ whose output we aim to explain for a specific task on a given dataset. For each input instance $d$, we generate a set $D'$ consisting of slightly perturbed versions of $d$. For instance, if $d$ is a text document, $D'$ might be created by randomly deleting words from various positions in $d$. The goal is to train a simple classifier $S$ that closely approximates $C$'s behavior on $D'$. Since no assumptions are made about $C$, $S$ is model-agnostic. Moreover, $S$ is local because it replicates $C$'s behavior for the given instances. Generally,

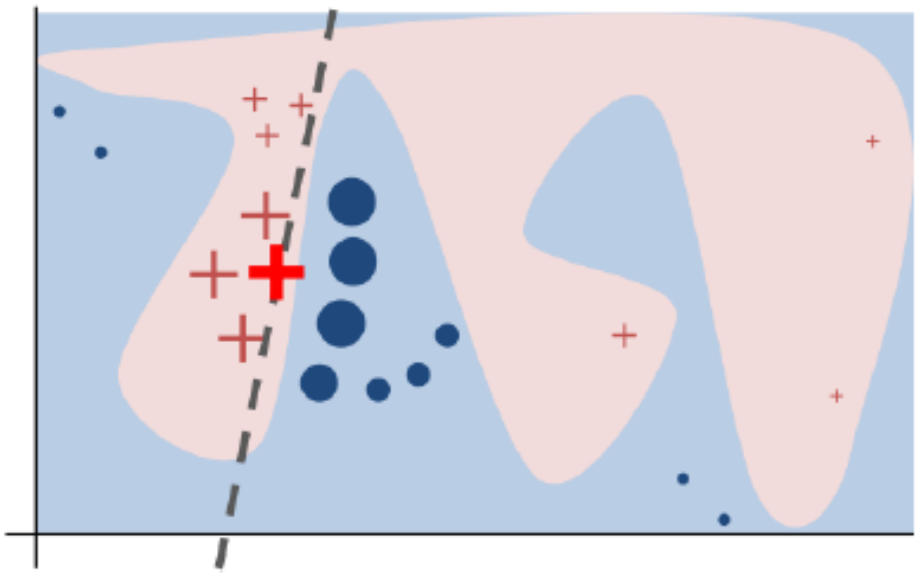


FIGURE 2.1: Schematic diagram of how LIME proposes to approximate the behaviour of a complex model locally via a set of perturbed instances. Image from the LIME paper [12].

no claim is made about the similarity between $C$ and $S$ across the entire input space of $C$.

Figure 2.1 shows the LIME framework schematically. The neural model has a complex and non-linear decision boundary. For a given instance, perturbations are made and labels are generated for these perturbations using the model. A simple linear model is then trained to classify this generated dataset of perturbed instances.

### 2.2.2 Feature Attribution

Feature attribution is a technique used to determine the contribution of each input feature to the predictions made by a machine learning model. It helps to understand which features are most influential in the model's decision-making process.

**Occlusion** is a method used in feature attribution to understand the importance of individual features in a model's prediction. In this technique, parts of the input data are systematically masked or occluded, and the model's output is observed to see how it changes. By measuring the impact on the prediction when a feature or group of features is occluded, one can infer the contribution of those features to the model's decision.

## 2.3 Explainable Search (EXS)

EXS [15] is an adaptation of the previously discussed LIME methodology designed to explain the output of a pointwise ranker for a given query. The main challenge in LIME is obtaining labels for the perturbed documents. While this task is straightforward for classification models, it is more complex for ranking models. EXS proposes three methods to acquire these labels.

Consider a query $q$ and let $D_q^k$ be the set of top-$k$ ranked documents provided by the complex reranking model $\mathcal{R}$. We need to determine whether a perturbed document $d'$ is relevant to build a sparse linear model that locally approximates the reranking model around the base document $d \in D_q^k$.

Let $X$ be a binary random variable indicating the relevance of $d'$ given the query $q$. We aim to estimate

$$P(X = \text{relevant}|q, d', \mathcal{R})$$

**Top-k Binary:** Here, we assume that $P(X = \text{relevant} \mid q, d', R) = 1$ if $R(q, d')$ exceeds $R(q, d'_k)$, where $d'_k$ is the $k$-th document in the list.

**Score-based:** In this scenario, we calculate $P(X = \text{relevant} \mid q, d', R)$ as:

$$1 - \frac{R(q, d_1) - R(q, d')}{R(q, d_1)}$$

where $d_1$ is the top-ranked document in the list. If $R(q, d') \geq R(q, d_1)$, $P(X = \text{relevant}) = 1$.

**Rank-based:** Here, we consider the rank of $d'$ in $D_q^k$. If $R(q, d') \leq R(q, d_k)$, $P(X = \text{relevant}) = 0$. Otherwise, $1 - \frac{\text{rank}(d')}{k}$, where $\text{rank}(d')$ is the rank of the perturbed $d'$ in $D_q^k$.

## 2.4 LIRME

LIRME [17] is another adaptation of LIME but in this method, we do not investigate ways of transforming document scores into class probabilities as in the case of EXS above. The sparse explanation models can be trained directly to predict the scores assigned by any ranking model, using the perturbed dataset.

By default, a Ridge regression model is fitted to the perturbed dataset, to predict the relevance scores of the perturbed documents, for the given query. The feature space is formed by the terms present in the document. Once the

model is fitted, the coefficients obtained from the regression model form the respective explanation scores of each term.

# Chapter 3

# The `ir_explain` library

`ir_explain` [14] is an open-source Python library[1] which incorporates several established techniques for Explainable Information Retrieval. It encompasses the standard categories of post-hoc explanations, including pointwise, pairwise, and listwise explanations. My contribution to this library has focused on integrating the concepts of pointwise explainability discussed in earlier chapters. This library equips users with multiple methods to generate pointwise explanations, along with features for visualization and evaluation of these explanations. Subsequent sections will explore the functionalities offered by the library, emphasising pointwise explanations.

## 3.1 Perturbed document generator

Perturbation and sampling is a key aspect of LIME, crucial for understanding how models make specific predictions. The EXS and LIRME methods provide different sampling strategies. All these strategies are combined in the `ir_explain` library in a single class named **PerturbDocument**. The class has three different methods corresponding to the different strategies. Any one of these methods can be used to produce the perturbed dataset.

### 3.1.1 Random sampler

This strategy takes the original document and randomly selects and removes words independently from the original document text. The number of words removed is also random.

[1] https://github.com/souravsaha/ir_explain

### 3.1.2 Masking sampler

In this strategy, the original document text is divided into a number of chunks of a given size. Each of the chunks are then present in the perturbed document with a given probability. Both the chunk size and the probability of any chunk being present are parameters which can be changed by the user.

### 3.1.3 Term frequency sampler

Term frequency sampler builds the perturbed document based on the term frequencies of the terms present in the original document. It uses the term-frequency data from the index itself for a given document. The index stores the terms of the document after performing stemming and lemmatization, along with their term frequencies. Out of all the $n$ terms stored in the index, the method randomly samples some $k$ number of terms, say $t_1, t_2, ..., t_k$. Let each of these terms have term-frequencies $f_1, f_2, ..., f_k$. The perturbed document has each of these k terms with the term $t_i$ present $f_i$ number of times. The number of terms sampled, $k$, is again a parameter which can be set by the user.

## 3.2 Explanation Generator

The `ir_explain` library implements the EXS and LIRME methodologies in two separate classes. These classes can instantiated with their respective set of parameters. Sample usage of these classes are shown below.

### 3.2.1 LIRME

The LIRME class needs to be instantiated with *index_path*, the path where the index is stored, from which information regarding the documents are to be fetched.

The *explain* method is used to generate the explanations. The query, document-id, document-score and the reranker model needs to be passed to the method. Other parameters required for generating explanations, for example, which sampling strategy is to be used, or how many terms to keep in the explanations, are passed as a dictionary, as shown in the code snippet.

```
# set parameter for LIRME
params = {
  "sampling_method" : "random",     # sampling method
  "top-terms" : 10,                 # num of explanation terms
  ...
}
```

```
# Instantiate object of LIRME explainer
lirme_explainer = LIRME(index_path)
# generate explanation vector
explanation_vectors, ranked_lists = lirme.explain(query, doc_id, \
    doc_score, reranker, params)
```

### 3.2.2 EXS

The EXS class needs to be instantiated with the reranker model and the type of linear classification model to be fitted, i.e. either svm or logistic regression. Based on this parameter, the corresponding linear classification model is used to fit the perturbed dataset and obtain term explanation scores, as described in section **2.3**.

Similar to LIRME above, *explain* method is used to generate explanations for the document at some given rank.

```
# Instantiate object of EXS explainer
exs_explainer = ExplainableSearch(reranker, "svm", num_samples=100)
#The rank of the document that needs to be explained
rank = 2
# generate explanation vector
explanation = explainer.explain(query, doc_ids, \
                rerank_scores, rank, doc_at_r, Method="topk-bin")
```

## 3.3 Evaluation metrics

In general, it is difficult to measure how good the explanation is, mostly because of the lack of ground truth data. Benchmark collections of IR do not provide any annotations for explanations. In literature researchers often use human evaluation to quantify the evaluation of explanations.

### 3.3.1 Proposed in LIRME

LIRME proposes two evaluation metrics for the explanations, namely, *correctness* and *consistency*. These involve first generating a *ground truth vector*. More details are provided in the following sections.

#### Ground truth vector

To generate the ground truth vector, we make use of benchmark datasets, like TREC and MS MARCO collections. These datasets contain benchmark queries with corresponding relevant documents. For any such benchmark query, we go through its set of relevant documents. For each unique term in this entire set

of relevant documents, we assign it an explanation score by averaging out its LMJM scores over the documents in which the term occurs. This forms the ground truth explanation vector for the given benchmark query.

An obvious drawback of this method is the requirement of such a benchmark dataset. Without these benchmark queries and the set of documents marked relevant for the query, above vector cannot be computed. The authors do not mention why LMJM based scoring was used to sample ground truth terms.

#### Correctness

Correctness of an explantion generated using any one of the pointwise techniques can be computed as the *similarity* between the explanation vector generated and the ground truth vector generated above. Cosine similarity is one such similarity measure which can be used to compute correctness.

#### Consistency

For a given query-document pair for which we want to generate an explanation, it is intuitively desirable that the explanations do not differ much if we change the parameters of the aforementioned explain functions. This can be quantified by the *consistency* of explanations which measures the correlation between the ordered list of explanation terms. LIRME proposes the use of Kendall tau rank correlation coefficient.

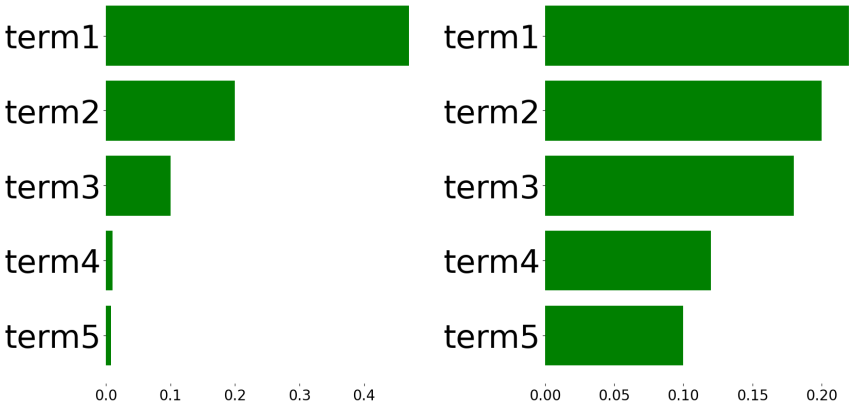
Such correlation can also be measured between a generated explanation and the ground truth vector to see how similar these two are. One difficulty here is that the ground truth vector can have terms not present in the generated explanation which makes computing kendall tau difficult.

### 3.3.2 Proposed metrics

We propose two evaluation metrics, i) occlusion based and ii) explanation score based measures to quantify explanations produced by pointwise explanation methods.

#### Occlusion based ground truth

For any arbitrary query-document pair, and a ranking model which assigns a relevance score to this pair, we can measure the importance of each term in the document by masking all occurrences of it and taking the change in relevance score as its importance. If a term has multiple occurrences, masking




(A) some terms more important (B) all terms have similar importance

FIGURE 3.1: A toy example showing two explanations with all terms having the same rank but very different scores. These explanations have perfect ranking correlation but are very different in relative importance of terms.

all occurrences will have greater impact. Hence we divide the change in relevance score by the number of occurrences of a term. This gives us the occlusion based ground truth, without the need of benchmark queries and relevant document datasets.

### Explanation score based dis-similarity

In the correlation based consistency metric, we do not consider the explanation scores of the terms. Only the relative order of the terms is taken into account. However, given the same order, the respective scores can vary significantly. Figure 3.1 shows a toy example consisting of just five terms, say term1, term2, ..., term5. The figure on the left shows the top terms having relatively higher scores in comparison to term4 and term5. On the other hand, the figure on the right shows all these terms having similar scores. Although the ranked order of the terms is same in both explanations, the relative importance of terms are quite different. A ranking correlation metric fails to capture this difference.

Given two term vector explanations, $E_1$ and $E_2$, let us consider there are $m$ common terms in both of them. Let these be $t_1, t_2, \ldots, t_m$ with explanation scores $e_{11}, e_{12}, \ldots, e_{1m}$ and $e_{21}, e_{22}, \ldots, e_{2m}$ in the two vectors. The dis-similarity between explanation scores for any term can be measured by the absolute difference in the two scores. We can then sum these absolute differences in scores

across the two explanations for each of these $m$ terms to get the total dis-similarity for these two explanations. Thus, we can define,

$$D(E_1, E_2) = \sum_{i=1}^{m} |e_{1i} - e_{2i}|$$

Important thing to consider here is the number of common terms $m$. If $m$ is low, it automatically implies higher dis-similarity between the two explanations as they have very distinct terms and vice versa. To incorporate this observation, we include a factor of $r = \frac{m}{N}$, where N is the total number of unique terms across the two explanations, to get the final dis-similarity measure. If $m = 0$, the measure is obviously undefined since there are no common terms. Thus,

$$D(E_1, E_2) = \frac{1}{r} \sum_{i=1}^{m} |e_{1i} - e_{2i}|$$

Using this measure along with the correlation metric, we can have better comparisons between two explanation vectors.

# Chapter 4

# Experiments

For the experiments carried out in this chapter, we have used the index **msmarco-v1-passage-full**, unless otherwise specified. The queries for the experiments are taken from the set of benchmark queries for this dataset. The following sections provide more details on the dataset and the queries used.

## 4.1 Usage of `ir_explain`

In this section we show an end to end usage of the `ir_explain` library for pointwise explanations. First documents are retrieved and re-ranked based on the two stage pipeline described in section 1.4 for a particular query. The top document according to the reranker is selected and explanations are generated using EXS and LIRME and visualizations for these explanations are shown. Evaluation metrics are then computed as described in the previous chapter.

### 4.1.1 Retrieve and rerank:

As dicussed previously, modern IR often involves a two stage pipeline − using a sparse retrieval model to fetch top-$k$ documents, followed by reranking of these top-$k$ documents by a complex neural model.

The query used is *"what is the daily life of thai people"*. This is one of the benchmark queries in the *MS MARCO passage collection* with query-id 1112341.

The function *get_results_from_index* in the `ir_explain` library uses the pyserini[1] module to search a given index. By default *BM25 similarity* is used to fetch documents from the index in pyserini. The *num_docs* parameter can be used to specify the number of documents to be retrieved in the first stage of the pipeline. The **BEIR** [16] python module is used to instantiate a neural

[1] https://pypi.org/project/pyserini/

reranker. We have used an SBERT based cross-encoder model named **ms-marco-electra-base**. The code is shown below.

```
#Use pyserini to search the index "msmarco-v1-passage-full"
searcher = LuceneSearcher.from_prebuilt_index("msmarco-v1-passage-full")

query = "what is the daily life of thai people"

retrieved_dict = get_results_from_index(query, searcher, num_docs=10)
doc_ids = retrieved_dict["doc_ids"]
docs = retrieved_dict["docs"]

# Load a reranking model
from beir.reranking.models import CrossEncoder
model = "cross-encoder/ms-marco-electra-base"
reranker = CrossEncoder(model)

#Rerank using the Neural model
rerank_scores = reranker.predict(list(zip([query]*len(doc_ids),docs)), batch_size=10)
retrieved_dict["rerank_scores"] = rerank_scores

print(rerank_scores)
```

The table 4.1 shows the list of documents retrieved along with their scores at the two stages of the pipeline. The document at the $5^{th}$ rank is given the highest score in the second stage of reranking.

| doc_id | Sparse retrieval score | Reranked score |
|---|---|---|
| **8139256** | **10.47** | 0.97951 |
| **2096429** | 10.18 | 0.00002 |
| **6740558** | 10.10 | 0.00002 |
| **8139258** | 10.01 | 0.98405 |
| **8139255** | 9.97 | **0.98601** |
| **2096427** | 9.32 | 0.00003 |
| **90432** | 9.29 | 0.00040 |
| **2735215** | 9.16 | 0.00006 |
| **6016292** | 9.11 | 0.00003 |
| **6164791** | 8.96 | 0.00052 |

TABLE 4.1: Results of the retrieve and rerank pipeline. Top score in each stage of the ranking is marked in bold.

We select the document with docid: 8139255 which is ranked highest by the re-ranker, for further analysis. The content of this document is shown below.

*"An important thing in everyday life is SANUK. Thai people love to have fun together. SANUK can represent many things : eat together, to be with friends and chat, to go out with friends. For Thai people SANUK happens with several persons."*

### 4.1.2 EXS explanation and visualization

We now use the EXS class to generate explanations for the document at rank 1. Note that, in the re-ranking stage, we are not removing the stopwords as we are using neural ranking model. Since the document contains stopwords, the generated explanation also contains these stopwords. We have a method named *remove_stopwords_from_explanation* which can be used to remove these stopwords from the explanations.

```
explainer = ExplainableSearch(reranker, "svm", 100)

rank = 1
doc_at_r = retrieved_dict["docs"][np.argsort(retrieved_dict["rerank_scores"])][-rank]

results = explainer.explain(query, doc_ids, rerank_scores, \
                    rank, doc_at_r, Method = "topk-bin")
#remove stopwords
results_no_stopword = explainer.remove_stopwords_from_explanation(\
                        results, stopword_file_path = "stop.txt")
#Visualize
explainer.visualize(list(zip(results_no_stopword[query][0], \
                    results_no_stopword[query][1])))
```

The table 4.2 shows the top-10 explanation terms with their scores after removing the stopwords. The figure 4.1 shows the visualization as well.

| S.No. | Term | Explanation score |
|---|---|---|
| **1** | life | 12.80 |
| **2** | thai | 11.54 |
| **3** | fun | 9.95 |
| **4** | contents | 5.60 |
| **5** | thing | 4.48 |
| **6** | people | 4.19 |
| **7** | eat | 2.98 |
| **8** | 8139255 | 1.22 |
| **9** | persons | 0.62 |
| **10** | things | 0.54 |

TABLE 4.2: Top-10 explanation terms with their explanation scores according to EXS.

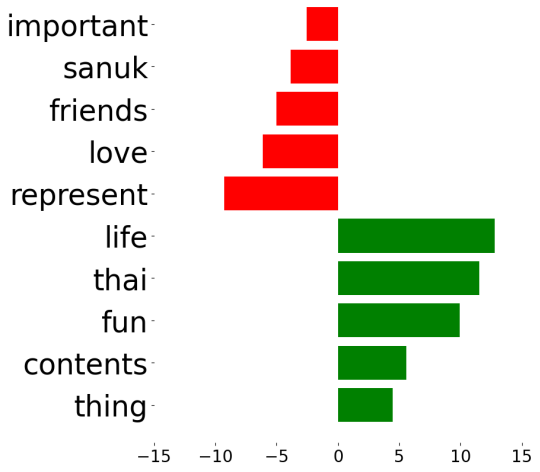



FIGURE 4.1: EXS term vector visualization after removing stopwords. Horizontal bars represent score of the term, with green denoting positive score and red, negative. Starting from the middle, we have top-5 positive and top-5 negative terms listed downwards and upwards respectively.

### 4.1.3 LIRME explanation and visualization

The LIRME class is used to generate explanations for the same document above. The visualization function can be used to show only positively influencing terms as well,using the parameter *show_pos_only*. This is demonstrated in Figure 4.2, which shows the top-5 positively influencing terms generated by LIRME.

```
query = "what is the daily life of thai people"

doc_id = "8139255"
doc_score = 0.98600733

params = {
    "sampling_method" : "masking",
    "top_terms" : 20,
    "kernel_range" : [5,10]

}

index_path = "/path/to/index/stored/"

lirme = Lirme(index_path)

explanation_vectors, ranked_lists = lirme.explain(query, doc_id, \
                                    doc_score, reranker, params)

lirme.visualize(explanation_vectors[0]["term_vector"], show_top = 5, show_pos_only=True)
```

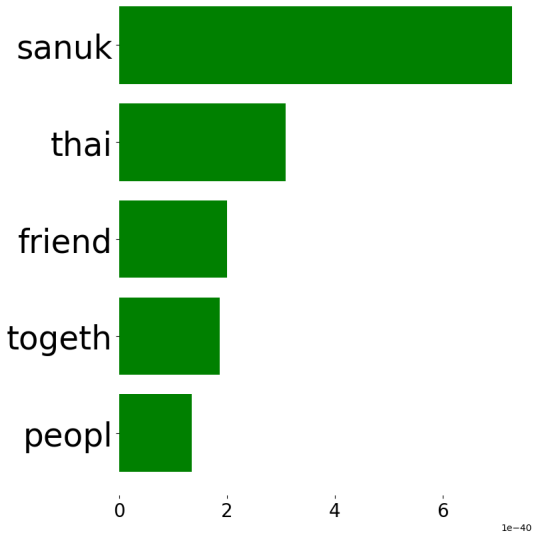



FIGURE 4.2: LIRME term vector visualization. Only two terms *eat* and *can* were given negative scores, which were relatively very low. Only the top-5 terms with positive scores are shown.

### 4.1.4 Evaluation:

```
query_id="1112341"
rel_docs_list = lirme.find_relevant_docs(query_id, num_rel_docs=10, \
                    dataset_name = "msmarco-passage/trec-dl-hard")

relevant_doc_vector = lirme.generate_ground_truth_terms(rel_docs_list)


lirme.correctness(doc_id, explanation_vectors[0]["term_vector"], relevant_doc_vector)
```

#### Relevant-doc Vector

For the given query, we fetch the the set of documents marked relevant from the benchmark dataset *msmarco-passage/trec-dl-hard* using the python module *ir_datasets* [8]. Using these relevant documents, we create the ground truth vector as described in LIRME. Table 4.3 shows the term vector obtained and Figure 4.3 shows the visualization of the same.

| S.No. | term | explanation score |
|---|---|---|
| **1** | thai | 0.73 |
| **2** | songkran | 0.35 |
| **3** | thailand | 0.34 |
| **4** | chiang | 0.21 |
| **5** | languag | 0.17 |
| **6** | à | 0.14 |
| **7** | siames | 0.13 |
| **8** | april | 0.13 |
| **9** | festiv | 0.12 |
| **10** | isan | 0.11 |

TABLE 4.3: Term vector obtained from the 10 relevant docs for the given query.

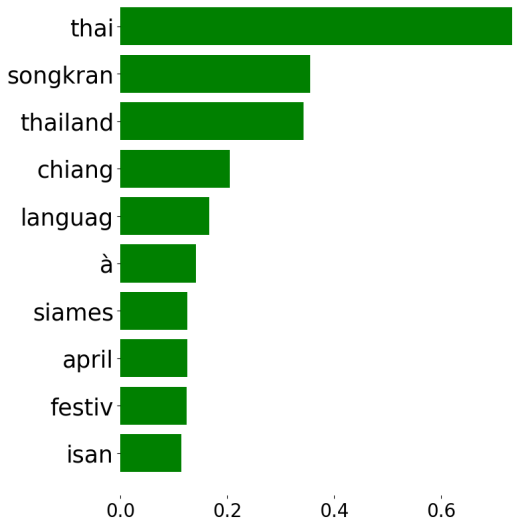



FIGURE 4.3: Term vector obtained from the 10 relevant docs for the given query.

**Correctness** The cosine similarity between the explanation generated and the ground truth vector comes out to be **0.25**

### 4.1.5 Occlusion based ground truth

Figure 4.4 shows the terms with their occlusion based importance scores. The top-5 and bottom-5 terms are shown in green and red respectively.

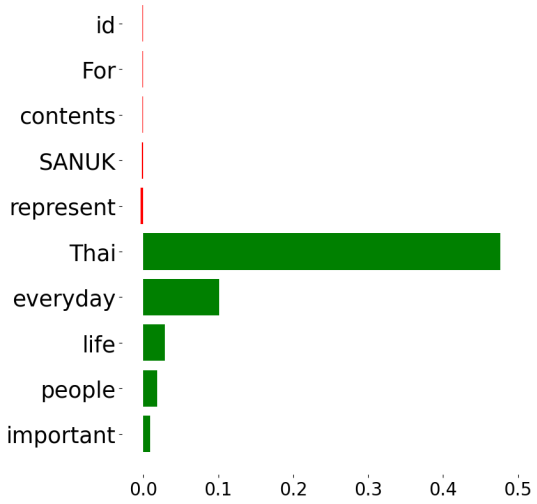



FIGURE 4.4: Term vector obtained by occluding all occurrences of each unique term, one by one.

### 4.1.6 Observations

- EXS considers ***life***, ***thai*** and *fun* as the top three terms, two of which are part of the query, marked in bold here.
- LIRME considers *sanuk*, ***life*** and ***thai*** as the top three terms, two of which are again part of the query, marked in bold.
- The ground truth vector considers ***thai***, *songkran* and *thailand* as the top three terms. Songkran is the Thai New Year.
- Occlusion based approach considers ***thai***, *everyday* and ***life*** as the top three terms. The next term is ***people***. All these terms can be considered to be part of the query or minor modification of it (if we look at everyday as a modification of *daily*).

Looking at the document text manually, SANUK is referred multiple times and seems to be important to the query. In fact, SANUK is used in English to describe the Thai cultural idea that values fun and enjoyment as "a regular and important component of everyday life" [6]. This makes the document highly relevant for the query in concern.

Both EXS and the occlusion based approach consider *sanuk* to be negatively influencing the relevance score. However, LIRME considers *sanuk* as the most important term.

One possible reason for the low score to *sanuk* by two of the methods above could be the way the reranking model *tokenizes* the input string.

```
#Using AutoTokenizer library from huggingface for the reranking model used
tokenizer = AutoTokenizer.from_pretrained("cross-encoder/ms-marco-electra-base")

query = "An important thing in everyday life is SANUK."
# Tokenize the text
tokenized_text = tokenizer.tokenize(query)
print(tokenized_text)

#Output:
["an", "important", "thing", "in", "everyday", "life", "is", "san", "##uk", "."]
```

As we can see above, *sanuk* is broken into two components, "san" and "##uk" by the tokenizer. This may lead to the low explanation scores as the ranking model does not indentify *sanuk* in the context of Thai culture. However, why LIRME considers it to be the most important word needs further analysis.

## 4.2 Robustness

In literature, Robustness of explanations, that is, explanations generated for *similar* input instances should be *similar*, is considered desirable. [1, 5] Intuitively, if the input being explained is altered slightly, just enough to avoid significantly affecting the model's prediction, we would expect that the explanation given by the interpretability method for this new input should remain largely unchanged. The *similarity* in explanations can be computed using the methods described previously, and indeed requires further work.

### 4.2.1 Manual perturbation example

Given the document D we have used above, we create $D'$ by making slight perturbations to D without changing the document's actual intent. Table 4.4 displays the contents of both documents, and Figures 4.1, 4.5 illustrate the explanation terms and weights generated by EXS for D and $D'$. The first notable observation is the significant difference in explanation terms between D and $D'$.

The term *life* positively influences both D and $D'$ the most, while *represent/represents* negatively influences both D and $D'$ the most. However, *thai* has negative score for $D'$.

It is evident that the explanations provided by EXS are unstable in this example. Although this single instance does not definitively prove that the explainer lacks robustness, it illustrates that `ir_explain` enables a more thorough analysis of such explainers.

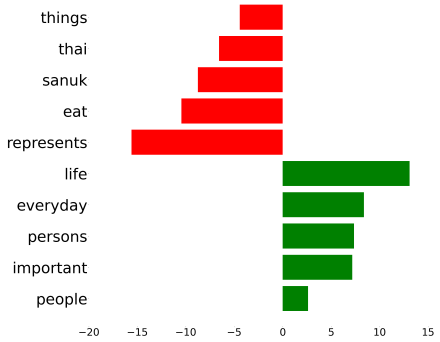



FIGURE 4.5: EXS term vector visualization after manually perturbing the original document D.

| *Query (qid: 1112341) what is the daily life of thai people* | | | |
|---|---|---|---|
| Rank | Document | Rel. | Content |
| 1 | $D$ (docid: 8139255) | 3 | An important thing in everyday life is SANUK. Thai people love to have fun together. SANUK can represent many things : eat together, to be with friends and chat, to go out with friends. For Thai people SANUK happens with several persons. |
| - | $D'$ (Perturbed) | - | An important thing in everyday life is SANUK. Thai people love to have fun together. SANUK ~~can~~ represents many things : eat together, to be with friends and chat, to go out with friends. For **most** Thai people SANUK happens with multiple persons. |

TABLE 4.4: The contents of the original and the manually perturbed document. In this example, $D$ is the original document and $D'$ is the perturbed instance of the document $D$. The changes made to $D$ are highlighted using strike-throughs, underlined and bold words for removal, modification and addition of words, respectively. It can be observed that there is only a slight difference between these two documents.

## 4.3 Consistency experiment

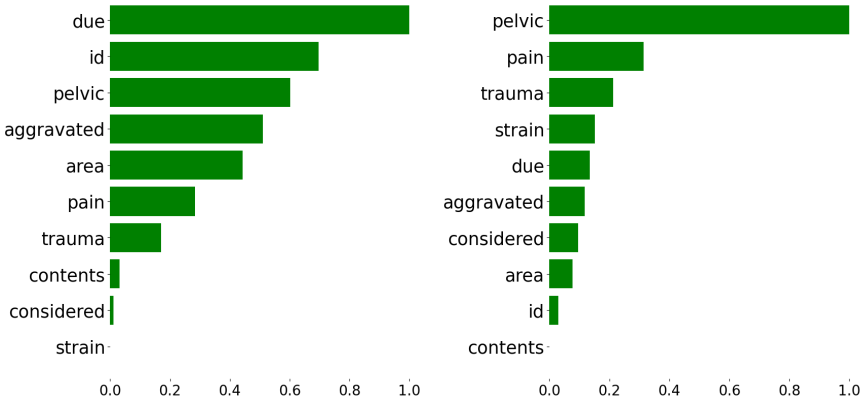
For a particular query-document pair, we first generate an explanation using EXS and consider the top-10 terms. For each of these top-10 terms, we calculate their occlusion based importance scores and rank the terms accordingly. The kendall tau correlation is computed between the order of terms according to the EXS explanation and occlusion. The score based dissimilarity is also calculated between the two, as described in section **3.3.2**.

This procedure is repeated for 10 benchmark queries from the *msmarco-passage/trec-dl-hard* dataset and the top-5 documents for each query. The results for a few of these query-document pairs are shown in table 4.5. The mean kendall tau value across all 10 queries with 5 documents each, comes out to be **0.16**. This indicates that the explanations generated by EXS are not very consistent with those generated by occlusion, on average. For query-id **156493** and document-id **3288600**, quite high kendall tau value of **0.69** is obtained. However, for the same query and document-id **1960255**, a negative value of **-0.16** is obtained.

To see the utility of the score based dissimilarity measure, let us consider the query-id **489204** and the documents **1778458** and **1051498** (towards the end of table 4.5). The kendall tau value for both is **0.07**. The normalized term vectors obtained by EXS and occlusion for documents 1778458 and 1051498 are shown in figures 4.6 and 4.7 respectively. We can see that the term-vectors in 4.6 have a much similar shape, with respect to relative importance of each of the top-10 terms, than that in 4.7. This is captured by the dissimilarity score of document 1051498 of **0.51** being much higher than that of document 1778458 of **0.30**.

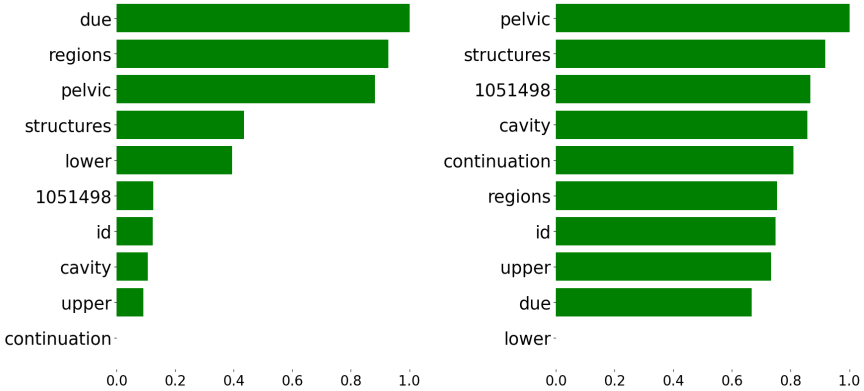
| **query_id** | **doc_id** | **kendall tau coefficient** | **explanation score based dis-similarity** |
|---|---|---|---|
| **156493** | 1960255 | -0.16 | 0.34 |
| **156493** | 3288600 | 0.69 | 0.51 |
| **156493** | 8182159 | 0.33 | 0.44 |
| **1110199** | 4511137 | 0.38 | 0.24 |
| **1110199** | 8160520 | -0.02 | 0.37 |
| **1110199** | 3838645 | 0.11 | 0.34 |
| **1063750** | 4788295 | -0.11 | 0.39 |
| **1063750** | 7778351 | 0.20 | 0.30 |
| **1063750** | 6093904 | 0.29 | 0.28 |
| **130510** | 799647 | 0.02 | 0.51 |
| **130510** | 1494936 | 0.07 | 0.35 |
| **130510** | 8612906 | 0.60 | 0.33 |
| **489204** | 1778458 | 0.07 | 0.30 |
| **489204** | 1051498 | 0.07 | 0.51 |
| **489204** | 8737051 | -0.16 | 0.38 |

TABLE 4.5: Consistency between EXS and Occlusion explanations. The scores for 5 benchmark queries, with top-3 documents for each query, are shown.




(A) EXS term vector (B) occlusion term vector

FIGURE 4.6: Term vectors obtained by EXS and occlusion as described in section 4.3, for the query "right pelvic pain causes" (query-id: 489204) and document id 1778458.




(A) EXS term vector (B) occlusion term vector

FIGURE 4.7: Term vectors obtained by EXS and occlusion as described in section 4.3, for the query "right pelvic pain causes" (query-id: 489204) and document id 1051498.

# Chapter 5

# Summary and Future Work

While numerous frameworks for Information Retrieval (IR) explanations exist, a unified framework that integrates these diverse approaches is uncommon. The `ir_explain` library addresses this gap by consolidating various explainability techniques into a single, cohesive module. This integration facilitates diverse and comprehensive analyses, as demonstrated in this thesis, by simplifying the application of multiple explanation methods. It is anticipated that `ir_explain` will encourage and enable further advancements in the field of IR explanations, promoting the development of more transparent and interpretable machine learning models.

## 5.1 Future Work

As the immediate next step, we discuss two possible future directions: i) evaluating pointwise explanations, and ii) making explanations robust and stable. In this section, we provide a brief overview of these.

### 5.1.1 Evaluation

As evident from several use cases described in this thesis, the evaluation metrics for explainability in Information Retrieval (IR) are insufficient. The current metrics do not comprehensively capture the effectiveness and reliability of explanations provided by different techniques. Future work should focus on developing and refining evaluation metrics that are better suited to assess the quality of IR explanations.

### 5.1.2 Robustness

Another critical area for future research is the robustness of explanation methods. Adversarial perturbations [18, 7] pose a significant challenge, as they can drastically alter the output of IR models and their explanations. Investigating the impact of such perturbations on explanation stability and reliability is essential. Future work should build on these insights to develop more resilient explanation techniques that can withstand adversarial attacks and maintain their integrity in various scenarios.

# Bibliography


[1] David Alvarez-Melis and Tommi S Jaakkola. "On the robustness of interpretability methods". In: *arXiv preprint arXiv:1806.08049* (2018).

[2] Avishek Anand et al. "Explainable information retrieval: A survey". In: *arXiv preprint arXiv:2211.02405* (2022).

[3] Paul Covington, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations". In: (2016), pp. 191–198.

[4] Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608* (2017).

[5] Leif Hancox-Li. "Robustness in machine learning explanations: does it matter?" In: (2020), pp. 640–647.

[6] Arne Kislenko. *Culture and Customs of Thailand*. Bloomsbury Publishing, 2004. ISBN: 9780313321283.

[7] Yu-An Liu et al. "Topic-oriented Adversarial Attacks against Black-box Neural Ranking Models". In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2023, pp. 1700–1709.

[8] Sean MacAvaney et al. "Simplified Data Wrangling with ir_datasets". In: *SIGIR*. 2021.

[9] Dmitrii Moor et al. "Exploiting Sequential Music Preferences via Optimisation-Based Sequencing". In: (2023), pp. 4759–4765.

[10] Manish Raghavan et al. "Mitigating Bias in Algorithmic Employment Screening: Evaluating Claims and Practices". In: *CoRR* abs/1906.09208 (2019). arXiv: 1906.09208. URL: http://arxiv.org/abs/1906.09208.

[11] Brian Regan, Desislava Hristova, and Mariano Beguerisse-Díaz. "Semi-Automated Music Catalog Curation Using Audio and Metadata". In: (2023).

[12] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proc.of SIGKDD 2016*. 2016, 1135–1144.

[13] Sourav Saha, Debapriyo Majumdar, and Mandar Mitra. "Explainability of Text Processing and Retrieval Methods: A Critical Survey". In: *arXiv preprint arXiv:2212.07126* (2022).

[14] Sourav Saha et al. "ir_explain: a Python Library of Explainable IR Methods". In: *arXiv preprint arXiv:2404.18546* (2024).

[15] Jaspreet Singh and Avishek Anand. "Exs: Explainable search using local model agnostic interpretability". In: (2019), pp. 770–773.

[16] Nandan Thakur et al. "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021. URL: https://openreview.net/forum?id=wCu6T5xFjeJ.

[17] Manisha Verma and Debasis Ganguly. "LIRME: locally interpretable ranking model explanation". In: (2019), pp. 1281–1284.

[18] Chen Wu et al. "Prada: Practical black-box adversarial attacks against neural ranking models". In: *ACM Transactions on Information Systems* 41.4 (2023), pp. 1–27.