ir_explain
a Python Library of Explainable IR Methods

Saha, Sourav; Agarwal, Harsh; Venktesh, V.; Anand, Avishek; Mohanty, Swastik; Majumdar, Debapriyo; Mitra, Mandar

DOI
10.1145/3726302.3730343

Publication date
2025

Document Version
Final published version

Published in
SIGIR 2025 - Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval

Check for updates

ir_explain: a Python Library of Explainable IR Methods

Sourav Saha
Indian Statistical Institute
Kolkata, India
sourav.saha_r@isical.ac.in

Harsh Agarwal
Indian Statistical Institute
Kolkata, India
1996harsh.agarwal@gmail.com

Venktesh V
Delft University of Technology
Delft, The Netherlands
V.Viswanathan-1@tudelft.nl

Avishek Anand
Delft University of Technology
Delft, The Netherlands
Avishek.Anand@tudelft.nl

Swastik Mohanty
Indian Statistical Institute
Kolkata, India
swastik.mohanty63@gmail.com

Debapriyo Majumdar
Indian Statistical Institute
Kolkata, India
debapriyo@isical.ac.in

Mandar Mitra
Indian Statistical Institute
Kolkata, India
mandar@isical.ac.in

Abstract

While recent advancements in Neural Ranking Models have resulted in significant improvements over traditional statistical retrieval models, it is generally acknowledged that the use of large neural architectures and the application of complex language models in Information Retrieval (IR) have reduced the transparency of retrieval methods. Consequently, Explainability and Interpretability have emerged as important research topics in IR. Several axiomatic and post-hoc explanation methods, as well as approaches that attempt to be interpretable-by-design, have been proposed. We present `ir_explain`, an open-source Python library that implements a variety of well-known techniques for Explainable IR (ExIR) within a common, extensible framework. It supports the three standard categories of post-hoc explanations, namely pointwise, pairwise, and listwise explanations. The library is designed to make it easy to reproduce state-of-the-art ExIR baselines on standard test collections, as well as to explore new approaches to explaining IR models and methods. To facilitate adoption, `ir_explain` is well-integrated with widely-used toolkits such as Pyserini, PyTerrier (work in progress) and `ir_datasets`. Downstream applications of `ir_explain` include explaining the Retrieval-Augmented Generation (RAG) pipeline. The development version of the library is available on GitHub. We release the library as a pip package (https://pypi.org/project/ir-explain/); source code is available from https://github.com/souravsaha/ir_explain.

CCS Concepts

• **Information systems → Information retrieval**.

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730343

Keywords

explainable information retrieval, post-hoc interpretability, interpretable by design, axiomatic ranking, probing

ACM Reference Format:
Sourav Saha, Harsh Agarwal, Venktesh V, Avishek Anand, Swastik Mohanty, Debapriyo Majumdar, and Mandar Mitra. 2025. `ir_explain`: a Python Library of Explainable IR Methods. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3726302.3730343

1 Introduction and motivation

Large Language Models (LLMs) [8, 9, 11, 28, 31, 40] are the most recent in a succession of increasingly complex deep learning models that have been applied to Information Retrieval (IR). These models are frequently almost magically effective at fulfilling users' information needs in a wide variety of tasks. At the same time, the models are also known to have important flaws, e.g., a tendency to 'hallucinate' in certain circumstances. If the output of such a system included (in addition to the final results) an *explanation* of how the system arrived at these results, an end-user would be better able to judge the trustworthiness of the answer. This would likely lead to increased adoption of these remarkable technologies in high-risk situations. Accordingly, the emphasis on 'understanding' these complex models has increased, and a good deal of research has been done in recent times on the explainability of Machine Learning (ML) models in general, and the so-called 'mechanistic interpretability' of LLMs in particular. Our focus in this work is on methods that are specific to Explainable IR (ExIR). While these methods are often inspired by ML explainability techniques, they have a distinct flavour and purpose because they explicitly deal with ranking rather than classification. A detailed overview of research in the broad area of ExIR can be found in [2, 3, 35]. The diverse research efforts summarised in these overviews constitute a good start, but more concerted attention is required from the community in order to address the many challenges that remain, before IR systems can explain their output in ways that are simultaneously precise, clear and intuitive.

Our contribution

In order to encourage new research in this area, we have put together `ir_explain`, a Python library of ExIR methods. To the best of our knowledge, this is the first library that implements all the major approaches that have been proposed so far to explain rankings produced by complex IR models. The target audience of `ir_explain` includes researchers working in ExIR who are looking for a convenient way to compare newly proposed approaches with existing ones, as well as practitioners who simply want to applying existing methods to 'debug' or explain neural ranking models (NRMs) that are used in practical applications. We hope this package will also reduce the barrier to entry for new researchers, and make it easy for them to explore all major existing ExIR techniques, and experiment with new ones.

`ir_explain` is intended to be well-integrated with widely-used toolkits such as Pyserini and PyTerrier, as well as `ir_datasets` for test collection management. The library is thus expected to eliminate the burden of (i) locating available ExIR implementations scattered across different locations, and (ii) setting them up to work with one's choice of an engine for running IR experiments.

`ir_explain` provides a framework for implementing and analysing new approaches to ExIR, and includes an evaluation component for measuring the fidelity of explanations via similarity measures commonly used for comparing IR rankings.

To showcase its capabilities, we use `ir_explain` in this study to (i) analyse the robustness of a pointwise explanation method, (ii) examine whether the explanations (lists of terms) produced by listwise methods are sufficiently intuitive, (iii) provide a usecase of pairwise explanation, (iv) study the replicability of previously reported experimental results obtained using listwise explainers, and (v) demonstrate `ir_explain` in the retrieval-augmented generation (RAG) pipeline. Source code for `ir_explain` is available from https://github.com/souravsaha/ir_explain. `ir_explain` is also available for installation from the Python Package Index (https://pypi.org/ir-explain/) using `pip install ir-explain`.

2 Related work

The IR community has a long tradition of providing open-source libraries and resources that make it convenient for researchers to experiment with, evaluate and understand different retrieval models. Anserini [47], Pyserini [19], PyTerrier [25] and PyGaggle [27] are examples of modern IR toolkits that support both sparse and dense retrieval models, and provide easy access to the standard retrieve-and-rerank pipeline on TREC collections.

Other well-known IR resources that are related to `ir_explain` include `ir_datasets` [24] (for acquiring and managing test collections for ad hoc IR) and `ir_axioms` [7]. The `ir_axioms` package implements various retrieval axioms [12], as well as a method for aggregating the document ordering preferences specified by different axioms into a single ranking [14]. To facilitate working with standard datasets and IR tools, `ir_axioms` is tightly integrated with `ir_datasets` and PyTerrier. The design of `ir_explain` is strongly influenced by `ir_axioms`. Indeed, `ir_explain` makes heavy use of `ir_axioms` in order to provide axiom-based explanations. Like `ir_axioms`, `ir_explain` is also tightly integrated with `ir_datasets`. However, while `ir_axioms` is integrated with PyTerrier, the current version of `ir_explain` makes use of Pyserini to access indexed collections. Integrating `ir_explain` with indices created by PyTerrier is work in progress.

While implementations of well-known ExIR approaches are not currently available within a single package, high quality interpretability libraries for ML models do exist. The AIX360 toolkit from IBM [4, 5] implements "diverse state-of-the-art explainability methods [including both global and local post-hoc explanations (e.g., LIME and SHAP), as well as 'direct' explanations], two evaluation metrics, and an extensible software architecture". Its primary focus is on binary classification, though regression is also addressed via Generalized Linear Rule Models [44].

Captum (https://captum.ai also implements many of the well-known interpretability techniques for ML models used in Computer Vision and Text Classification, e.g., feature attribution based on simple feature ablation, Integrated Gradients [39] or DeepLIFT [36], and LIME [34]. Captum is tightly integrated with PyTorch. While this provides easy access via `torchtext.datasets` to textual datasets like AG's News Corpus ([48], also see http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html) and the IMDb Large Movie Review Dataset [23], these collections are intended for tasks like text / sentiment classification, machine translation, question answering, and sequence tagging, rather than ranking and retrieval. Given the classification-oriented nature of AIX360 or Captum, and the level of integration between `ir_explain` and existing IR resources, we believe `ir_explain` is a much more natural fit for research and development in ExIR.

3 `ir_explain`: background and design

A two-stage pipeline is one of the standard paradigms used in modern Neural IR. First, given a query Q, a sparse retrieval model (e.g., BM25 or language models with Jelinek-Mercer (LMJM) or Dirichlet (LMDir) smoothing) is used to initially retrieve top k documents. The number k is chosen to be small compared to the collection size. In the second stage, a neural matching model (e.g., a cross-encoder or a dual-encoder) is employed to re-rank the set of k initially retrieved documents. Let us denote the first-stage ranker by M_1 and the second-stage ranker by M_2. The central problem of ExIR is to explain different aspects of these two stages of ranking.

While traditional, sparse retrieval models are generally regarded as *interpretable by design*, *post-hoc explanations* are most common in the case of neural ranking models (NRMs). These explanations may be classified into two groups based on whether they address (a) the document representation that is used to compute scores, or (b) the ranking decision induced by the model. Document representations generated by neural models are usually explained using *probing* methods, which study the lexical or semantic features that are encoded within these representations. Explanations of ranking decisions are in turn categorised as (i) pointwise, (ii) pairwise, and (iii) listwise explanations.

Figure 1 presents the overall structure of `ir_explain`, showing components that offer the various kinds of explanations listed above, along with auxiliary, supporting components. We briefly discuss the interpretable-by-design and probing components of `ir_explain` in Sections 3.4 and 3.5. In the rest of this section, we describe the

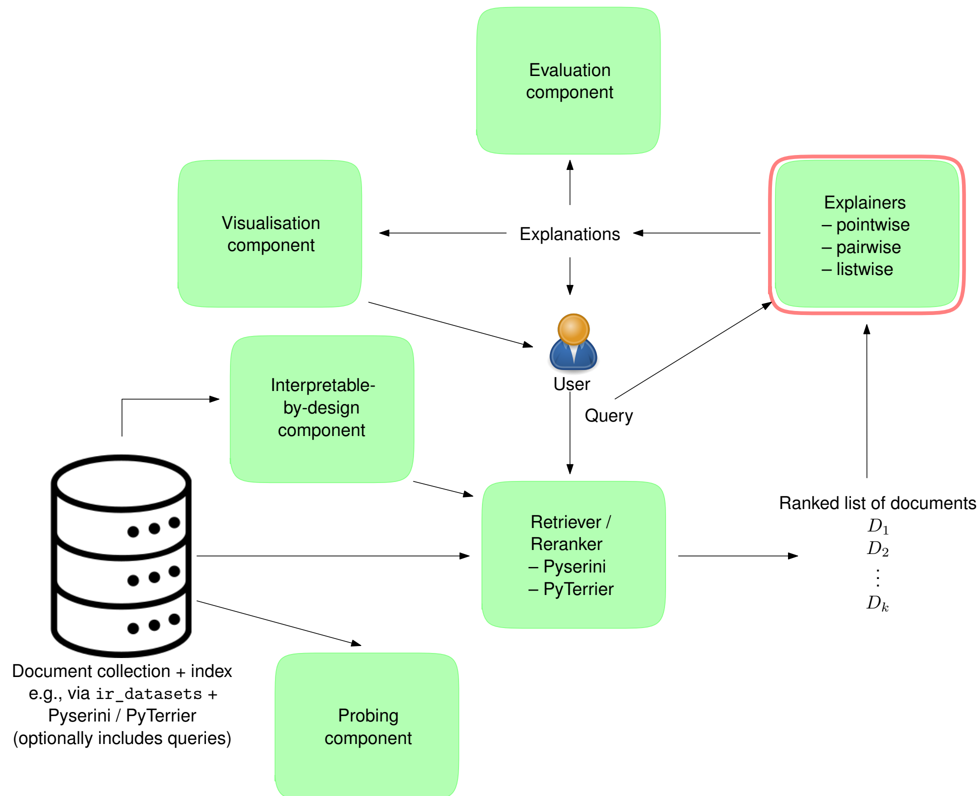


Figure 1: The overall structure of `ir_explain` showing how the various components of the library are intended to be used in practice.

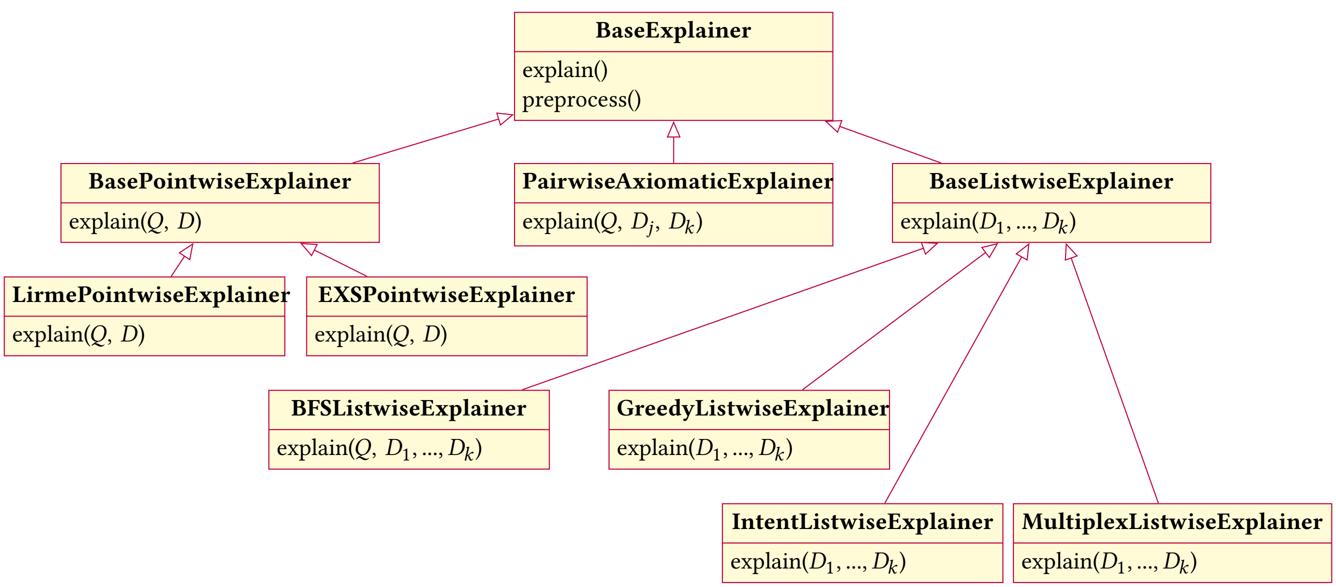


Figure 2: The inheritance class diagram of the different explanation modules used in `ir_explain` for explaining ranking decisions.

core component of `ir_explain` (highlighted in Figure 1), consisting of three major sub-modules, which implement pointwise, pairwise and listwise explanation methods. These sub-modules are designed so that they may be used to explain ranked lists generated on the fly (by specifying only a dataset, and any one of the dense retrieval, learned sparse retrieval, or hybrid retrieval models supported by Pyserini or PyTerrier). They may also be used to explain *any* ranked list (`runfile`) generated by leveraging an IR engine of one's choice, if such a list is supplied as an input to `ir_explain` via the `load_from_res` function.

3.1 Pointwise Explanations

In general, a pointwise explanation attempts to address two questions : (i) why is a particular document D retrieved within the top ranks by M_1? (ii) why does reranking by M_2 improve (or adversely affect) the rank of D? The explanations are typically in the form of a list of terms, along with weights that indicate how these terms influence the ranking of D, positively or negatively. Representative pointwise approaches include EXS [37] and LIRME [41], which uses ideas from LIME [34] within a document-ranking setup.

Figure 2 shows the inheritance relation between the various classes of explainers implemented in `ir_explain`. `BasePointwiseExplainer` inherits from the `BaseExplainer` class; in turn, `LirmePointwiseExplainer` and `EXSPointwiseExplainer` classes inherit from `BasePointwiseExplainer` class. One can easily add a new point-wise explanation approach by inheriting from this base class, and overloading the `explain` function. Listing 1 demonstrates how LIRME may be used to explain a particular query-document pair. The LIRME explainer is instantiated by passing it the index for a collection C; additional parameters such as `kernel`, `sampling_method`, and the number of desired explanation terms can be set through a dictionary. The `explain()` function of the `LirmePointwiseExplainer` class takes a query Q, a document D and additional parameters as input, and returns, as an *explanation vector*, a set of terms and their contributions (positive or negative) to the rank of D.

The `TermVisualization` class can be instantiated to store an explanation vector, and its `visualize()` function helps visualize the vector, as shown in Figure 4. The `eval` module of `ir_explain` helps to evaluate different types of explanations. `PointWiseCorrectness` and `PointWiseConsistency` classes and their `evaluate()` methods implement the ground-truth generation strategies proposed in LIRME. All the different sampling strategies proposed in both LIRME and EXS have been included.

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

# instantiate object of LIRME explainer
lirme = LirmePointwiseExplainer(model, corpus_path =
    "/path/to/index", indexer_type= "pyserini")

# generate explanation vector
explan_vector, ranked_list = lirme.explain(query,
    docid, params)

# visualize explanations
visualizer = TermVisualization()
visualizer.visualize(explan_vector, show_top=5)

# Evaluation component
# compute pointwise explanation correctness
correctness = PointWiseCorrectness(lirme)
correctness.evaluate(query, docid, explan_vector)

# compute pointwise explanation consistency
consistency = PointWiseConsistency(lirme)
consistency.evaluate(query, docid, ranked_list)
```

Listing 1: Example of a pointwise explainer LIRME with a query, document, and a set of parameters passed to it.

3.2 Pairwise Explanations

A pairwise explanation attempts to explain why a document D_i is preferred over another document D_j for a particular query Q. Such explanations are often provided in terms of retrieval axioms. Axioms are formalisations of intuitive retrieval heuristics that specify constraints that a good ranking method should fulfil. Several sets of axioms have been formulated in the literature [1, 12, 13], and a few relaxations have been proposed to make the axioms usable in practice [14]. `ir_explain` leverages the `ir_axioms` library for pairwise explanations. Following `ir_axioms`, each retrieval axiom corresponds to a separate Python class in `ir_explain`.[1] Recall from [7] that, given $\langle Q, D_i, D_j \rangle$, an object corresponding to axiom A returns a preference score $pref_A(Q, D_i, D_j) \in \{-1, 0, +1\}$ that specifies whether D_i is preferred over D_j by A: a score of $+1$ (resp. -1) signifies D_i is (resp. not) preferred over D_j, and a 0 means no preference. The implementation in `ir_axioms` is, however, tightly integrated with the Pyterrier retrieval pipeline, and does not seem to have a provision for explaining an arbitrary document pair $\langle D_1, D_2 \rangle$; this feature has been included in `ir_explain`. As in `ir_axioms`, users can use binary and unary operators to aggregate different axioms, and may also easily define new axioms. In `ir_explain`, we have segregated all the axioms into a separate pairwise explanation module.

Listing 2 demonstrates how this module works in `ir_explain`. The `PairwiseAxiomaticExplainer` class is instantiated with $\langle Q, D_i, D_j \rangle$ and the path to a collection index. This class is inherited from the `BaseExplainer` class. The `explain()` function takes a list of `axioms` for which the preference needs to be computed. For more involved axioms (such as `PROX1`, `PROX2`, `PROX3` [7]), the preference score alone does not adequately explain why one document is preferred over another. The function `explain_details()` (new in `ir_explain`) may be used to obtain a more detailed understanding in such cases. For example, according to the `PROX1` axiom, a document D_1 is preferred over D_2 if D_1 has a shorter average distance between the query terms. Table 1 shows the output of `explain_details()` for $\langle Q, D_i, D_j \rangle$, which compares the raw term frequencies (`tf`) of query terms, as well as the average distance between the query tokens in D_i and D_j.

3.3 Listwise Explanations

Listwise approaches provide a single explanation for an entire ranked list $L = \{D_1, D_2, ..., D_k\}$. These explanations are typically in the form of a pair $\langle Q_{exp}, SM \rangle$, where Q_{exp} is an expanded version of the query Q, and SM is a 'simpler' ranker, such as BM25 that is used

[1] The `Argumentativeness` axioms have been omitted for now, since they do not apply to the ad-hoc retrieval settings used in our experiments.

```
# instantiate pairwise object
pairwise = PairwiseAxiomaticExplainer(query, doc1,
    doc2, index_path)

# instances of various axiom classes
axiom_classes = [TFC1(), STMC1(), ...]

# call the explain method
pairwise.explain(axiom_classes)

# call the explain_details method to understand further
pariwise.explain_details(query, doc1, doc2, axiomName)
```

Listing 2: Example use-case of a pairwise explanation module to explain a query-document pair. A list of axiom objects can be passed to the explain function.

Table 1: Output of `explain_details` method for the query *'exons definition biology' (qid: 183378)* and a pair of documents (`D1077802 and D1806793`). These two documents are sampled from the MS MARCO document collection.

Query: exons definition biology (qid: 183378)		
docid	D1077802	D1806793
tf(exon)	23	21
tf(definit)	7	56
tf(biolog)	1	25
avg_dist(exon, definit)	174.43	2728.07
avg_dist(definit, biolog)	354.71	3287.24
avg_dist(exon, biolog)	315.04	2864.24
num pairs	3	3
Total_avg_dist	281.39	2959.85

to approximate a neural model. The interpretation of this explanation is that, for Q, the complex ranker M_2 behaves as if it implicitly finds the same matches that SM would explicitly find, given Q_{exp}. The task of finding an explanation reduces to (i) constructing Q_{exp}, and (ii) verifying that the ranked list generated using $\langle Q_{exp}, SM\rangle$ is approximately equivalent to L. Based on how these sub-tasks are accomplished, listwise approaches can be divided into two broad categories. The first type considers L as a combination of multiple ordered pairs of documents, and seeks to explain as many of these pairs as possible. Thus, a list $\mathscr{L} = \{(D_j > D_k) : D_j, D_k \in L\}$ of preference pairs is first constructed by sampling document pairs from L. The next objective is to construct a Q_{exp}, such that $\langle Q_{exp}, SM\rangle$ preserves the ordering of most pairs of $\mathscr{L}$. The underlying hypothesis is that if SM agrees with M_2 on most of the randomly selected preference pairs, it is consistent with the full ranked list. Multiplex [22] and IntentEXS [38] belong to this category. The second class of approaches, exemplified by BFS [21] and Greedy [21], directly measure the similarity between L and the list returned using $\langle Q_{exp}, SM\rangle$, without decomposing L into pairwise preferences.

`ir_explain` implements four state-of-the-art listwise methods:

- `MultiplexListwiseExplainer`,
- `IntentListwiseExplainer`,
- `BFSListwiseExplainer`,
- `GreedyListwiseExplainer`.

```
# set parameters for Multiplex
params = {
  "max_k" : 10,                      # max exp. terms
  "min_k" : 4,                       # max exp. terms
  "EXP_model" : "language_model",    # explainer type
  "optimize_method" : "geno",        # optim. method
  ...
  }

# instantiate multiplex and call explain function
multiplex = MultiplexListwiseExplainer(index_path =
    "/path/to/index", indexer_type = "pyserini")

# invoke the explain method
multiplex.explain(qid, query_str, params)
```

Listing 3: Example of a listwise explainer Multiplex with a query, ranked list, and a set of parameters passed to the explainer.

Each of the corresponding classes is inherited from the `BaseListwiseExplainer` class (see Figure 2). Listing 3 shows an example of how `MultiplexListwiseExplainer` can be used to obtain an explanation. The `multiplex` object is an instance of the corresponding listwise class. Parameters such as the type of the simple explainer, document pair sampling strategy, optimization method, etc. are set via `dictionary`. The `explain()` function internally invokes several stages of the Multiplex explainer. We also support batch processing with the `explain_all()` function, which calls the `explain()` method for all the queries in a specified dataset. Additionally, `ir_explain` provides a few utility functions, e.g., `generate_candidates()`, and `show_matrix()`, that provide white-box views into various stages of the `explain()` method. Multiplex uses three simple rankers as explainers, whereas IntentEXS uses only one to explain a ranked list. `ir_explain` provides an opportunity to use IntentEXS with any one of the three simple explainers used in Multiplex. We have also augmented the list of simple explainers to include other widely used statistical retrieval models (such as LMDir and LMJM) in addition to the single model (BM25) used for experiments by Lyu and Anand [22]. Users can vary parameters such as the strategy for sampling document pairs, and there is a provision to use pluggable components for some parameters. Thus, a custom helper routine may be used to sample document pairs based on some specific distribution. `ir_explain` makes it convenient for the user to mix and match different explainers. One such use case is to adapt Multiplex in the framework of BFS. Further, in-built evaluation measures (briefly described in Section 3.6) such as Rank-Biased Overlap (RBO [43]) between the complex and simple ranker can be computed to compare the fidelity of the explanations provided by this variant of Multiplex, with baselines like BFS and Greedy.

3.4 Interpretable by Design

Although these approaches are not as popular as post-hoc explanations, they attempt to construct models that are, in principle, explainable by design. Explanation methods of this type generally require retraining the ranking model with interpretability in mind. SELECT-AND-RANK [17] is a recently proposed method from this category. It first extracts a small subset of the sentences from each document, and ranks documents based solely on these extracted snippets. Thus, the explanations provided in this paradigm are essentially concise (and presumably more easily understood) 'summaries' of the documents. The proposed algorithms train the model in an end-to-end fashion. `ir_explain` provides an interpretable-by-design component to build a ranking model from scratch using this approach. Additionally, a pretrained model will be made available for easy adoption of this approach.

3.5 Probing

In this module of `ir_explain`, the main emphasis is on understanding the encoded representation of documents holistically. Specifically, probing components allow us to analyze what IR properties (such as lexical and semantic matching, etc.) are encoded in retrieval models. There are some ongoing research efforts in this direction. Wallat et al. [42] proposed a causal probing framework for a widely used dual encoder model to analyze its capability of capturing six properties, namely: semantic matching, lexical matching (BM25), coreference resolution, named entity recognition, question classification, and term matching. `ir_explain` implements this causal probing framework to facilitate extensive analysis of the different layers of dual encoders and the features encoded within them.

3.6 Evaluation

Evaluation metrics related to ExIR, such as correctness, consistency, and fidelity, are implemented in the `evaluation` module of `ir_explain`. As illustrated in Listing 1, the `PointWiseCorrectness` class measures how well the pointwise explanation terms correlate with the expanded terms sampled using LMJM. On the other hand, the consistency measure (`PointWiseConsistency`) quantifies the relative differences in explanation terms when different sampling methods are used. A more detailed discussion can be found in [35, 41]. For listwise explanations, metrics such as Jaccard similarity and RBO are used to measure the overlap between the ranked lists produced by a complex ranker and a simpler one. Intuitively, RBO quantifies the similarity between these two ranked lists, and attaches greater importance to commonalities at top ranks. The evaluation of IR explanations is constantly evolving, however; these modules will need to be updated as newer and more robust evaluation approaches emerge.

3.7 Visualization

We provide visualization in the form of raw terms and their contributions (see Figure 4) for now. Many general-purpose visuzalization tools are available, but our current focus is primarily on explaining retrieval models. However, we plan to integrate `ir_explain` with various visualization tools and support different output formats such as `html`, `json`, etc.

3.8 Utilities

We discuss several utilities provided by `ir_explain`. One such popular component is document perturbation. There are different strategies to perturb a document and generate the perturbed instance D'. We provide `RandomSampler`, `MaskingSampler`, and `TfIdfSampler`. `RandomSampler` is the simplest; it randomly removes words from a document. In contrast, `TfIdfSampler` samples words from a document based on their tf-idf weights. A more sophisticated sampling based techniques can be added easily by using these utils module.

Table 2: Content of a document and the perturbed variant (D') of it. New words added to D' are shown in bold, modified words are underlined, and words that are removed are struck out.

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~~ <u>represents</u> many things : eat together, to be with friends and chat, to go out with friends. For **most** Thai people SANUK happens with <u>multiple</u> persons.

4 Demonstration of Use Cases

As an all-in-one toolkit for ExIR methods, `ir_explain` can be used for various experiments and analyses.We demonstrate few use cases of `ir_explain` here: (i) robustness analysis of pointwise explanations, (ii) investigating whether listwise explanations are sufficiently intuitive, (iii) usecase of pairwise explanation component, (iv) reproducibility study with `ir_explain`, (v) demonstration with RAG pipeline. We have used the MS MARCO passage, document collections, the TREC DL 2019, 2020, and hard [26] topic sets to articulate these use cases. We show that interesting insights and diverse analysis can be obtained easily using `ir_explain`. We believe this package will inspire and aid further research in ExIR.

4.1 Robustness of Pointwise Explanation

This section presents an anecdotal analysis of the robustness of the pointwise explainer `EXS`. Specifically, we examine the following hypothesis: if two documents D and D' are very similar in their content, the pointwise explanations provided for them by an explainer should not differ much. As an example, we consider the query *'what is the daily life of thai people'* (qid: 1112341) in the MS MARCO passage collection. For this query, BM25 retrieves a document D (docid: 8139255, judged 'highly relevant') at the 5th position; reranking with SBERT [33] it to the top position. We use `EXS` from `ir_explain` to generate a pointwise explanation for D. We also construct D' by perturbing D slightly, without altering the actual intent of the document at all. Table 2 shows the contents of both the documents. Figure 4 shows the explanation terms and weights produced by `EXS` for both D and D'. The first striking observation is that the explanation terms for D and D' differ a lot. Out of 10 terms shown for each document, only 3 are common, namely `represent/represents`, `sanuk` and `important`. According to `EXS`, two versions of the same term (`represents` in D, and `represent` in D') influence the rankings very negatively. On the other hand, the term `important` is said to have a positive influence on D whereas its influence on D' is negative. Another observation is that `EXS` marks `` `sanuk' ``

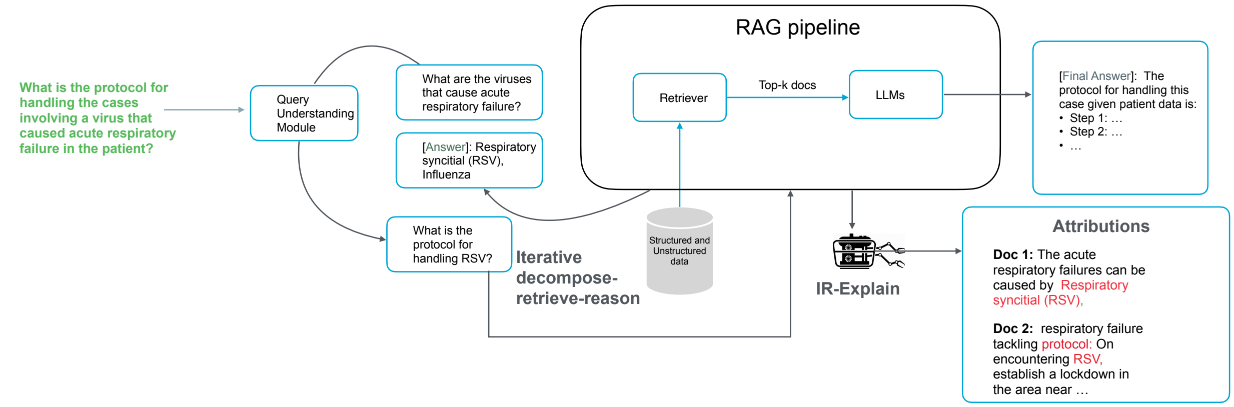
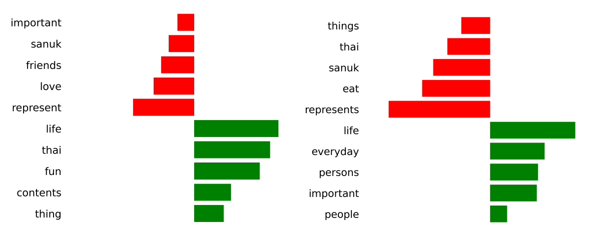


Figure 3: Example of how `ir_explain` can be used to detect salient terms for RAG attributions.

Table 3: A subset of results obtained by replicating experiments with BFS and Greedy based listwise explainers on TREC 2019 topic set. $MAP_{(rep.)}$ and $RBO_{(rep.)}$ represent our obtained values of MAP and RBO respectively (RBO values were computed using $p = 0.9$). IR models are grouped into sparse retrievals (Sparse), re-ranking-based dense retrieval (Rerank), and end-to-end dense (E2E) models. These three categories of models are similar to that in [21].

Model Type	IR Model	MAP	Greedy MAP	Greedy $MAP_{(rep.)}$	Greedy RBO	Greedy $RBO_{(rep.)}$	BFS MAP	BFS $MAP_{(rep.)}$	BFS RBO	BFS $RBO_{(rep.)}$
Sparse	BM25	0.1067	-	-	-	-	-	-	-	-
	RM3	0.1411	-	-	-	-	-	-	-	-
Rerank	DCT	0.2192	0.1544	0.1444	0.2207	0.2170	0.2050	0.1922	0.4946	0.4055
	QE_{BERT}	0.2199	0.1414	0.1382	0.2213	0.2208	0.2065	0.1852	0.5015	0.4107
E2E	ANCE	0.1836	0.1454	0.1393	0.2239	0.1982	0.1723	0.1744	0.4969	0.3815
	CBERT	0.2182	0.1506	0.1429	0.2230	0.2093	0.2064	0.1846	0.4888	0.4012
	MonoT5	0.2184	0.1470	0.1502	0.2224	0.2473	0.1920	0.1972	0.5194	0.4215



(a) Original document D (docid: 8139255) **(b) Perturbed document D' (from document D)**

Figure 4: Top-5 explanations terms of a sample document D and a slightly perturbed instance of D. The content of both the documents are shown in Table 2. We show the top-5 explanation terms, responsible for positively and negatively influencing the score of these documents. X-axis shows the magnitude of these terms and the Y-axis plots the explanation terms for the query *'what is the daily life of thai people' (TREC DL hard qid: 1112341).*

as a negatively influencing term in both instances. However, if we look at the content of the document (Table 2), `sanuk' means "having fun"; from a user's point of view, it is the most important word for this document. Clearly, the explanations provided by EXS are not stable for this example. While one example does not prove that a particular explainer is not robust, it does show `ir_explain` can be used easily for a more detailed analysis of explainers, that leads to interesting observations.

4.2 Approximating a Ranking as Explanation?

A listwise explainer claims to have explained a complex model M_2 if the ranked lists produced by M_2 and the explanation $\langle Q_{exp}, SM \rangle$ are 'close enough' according to some measure. However, the primary objective of generating *human-understandable* explanations gets overlooked if the success of an explainer is evaluated only in terms of rank correlation measures. Even when $\langle Q_{exp}, SM \rangle$ closely approximates M_2, the question remains: do the terms in Q_{exp} provide an intuitive explanation to the user? We show examples of listwise explanations provided by different approaches for two queries, and look at how well they relate intuitively to the query. Table 4 shows the explanations generated by BFS, Greedy, Multiplex, and IntentEXS explainers for two queries from the TREC DL hard query set when TCT-ColBERT [20] is used as the black-box neural ranking model. All of them approximate the ranking of TCT-ColBERT well, but for the first example (*'what is the daily life of thai people'*) the explanation terms generated by `IntentEXS` include many unrelated terms. A similar observation holds for `Greedy` in the second example (*'causes of stroke?'*): most terms do not convey the underlying intent of the query. Thus, from a user's point of view, these explanations are not intuitive, as they fail to explain the intent of the query. This highlights the need for a more carefully designed protocol for evaluating explanation strategies.

Table 4: Sample output of four different listwise explanation approaches generated on TREC DL hard queries. MS-MARCO passage collection is considered for the retrieval.

Explainer	Explanation terms
Query: what is the daily life of thai people (qid: 1112341)	
BFS	'thai', 'sanuk', 'temper', 'everydai', 'life'
Greedy	'sanuk', 'chat', 'friend', 'thing', 'togeth', 'happen', 'everydai', 'thai', 'life', 'fun'
Multiplex	'buddhist', 'thailand', '85,000', 'anger', 'life' 'daily', 'monk', 'people', 'what'
IntentEXS	'thai', '6.00', 'peopl', 'life', 'μà', 'put', 'though', 'do', 'vast', 'friend'
Query: causes of stroke? (qid: 88495)	
BFS	'causes', 'stroke', 'high', 'arteri'
Greedy	'uncommon', 'vast', 'harden', 'rarer', 'less', 'list', 'well', 'mani', 'follow', 'stroke'
Multiplex	'stroke', 'clot', 'causes', 'vast', 'die'
IntentEXS	'brain', 'arteri', 'list', 'usual'

4.3 Usecase of Pairwise Explanation

Recall from Section 3.3 that Multiplex and IntentEXS are designed to optimize for maximum preference pairs ($\mathscr{L}$). The output of these explainers, i.e., the explanation terms they produce, can also be used to provide a pairwise explanation induced by D_j and D_k. These explanations can be easily visualized in `ir_explain` by invoking the `show_matrix()` function of these listwise components and examining at the specific column of $\langle D_j, D_k \rangle$.

4.4 Replicability Study

With our library, replicating[2] baselines becomes a much easier task. As an example, we replicated the results produced by Llordes et al. [21] using `ir_explain`. Table 3 shows the results obtained using BFS and Greedy approaches on the TREC 2019 query set. The experiments were conducted on the same set of IR models (DCT [10], QE_{BERT} [21], ANCE [45], CBERT [16], MonoT5 [29]) as in [21]. We present a subset of the results and omit the Jaccard-based overlap measure; instead, we report RBO ($p = 0.9$), which appears to be preferred as a fidelity measure in recent literature. Obtained versions of MAP and RBO are denoted as $\text{MAP}_{(\text{rep.})}$ and $\text{RBO}_{(\text{rep.})}$ respectively. As observed from the table, the trends in the obtained figures are mostly similar to those reported by Llordes et al. [21]. However, for MonoT5-based ranking models, the Greedy baseline performs better than the reported figures, whereas in other cases, the $\text{RBO}_{(\text{rep.})}$ values are smaller. Though the randomness involved in their method is a potential source of these variations, a more thorough investigation is needed to determine the cause of this drop in performance. Note that we could not perform significance testing or compute replicability metrics due to the unavailability of the required details in Llordes et al. [21].

[2]We use this term and its variants in accordance with the definitions provided in https://www.acm.org/publications/policies/artifact-review-and-badging-current.

4.5 Demonstration with RAG Pipeline

`ir_explain` would aid in interpreting Retrieval-Augmented Generation (RAG) pipelines. While LLMs have made strides in a wide range of NLP tasks they still suffer from hallucination [15], where the LLM makes up plausible yet non-factual statements due to knowledge gaps. While RAG [18] aids in mitigating this issue to an extent the problem of attribution persists. Attribution entails determining if the answer generated by the LLM is faithfully supported by the retrieved documents. This task is difficult due to the large number of possible documents and large output space [30, 46]. While several answer attribution approaches have been proposed [6, 32], they still rely on an external validator, such as NLI (Natural language Inference) models. hence, such approaches cannot guarantee the faithfulness of the attribution process. Hence, more recently approaches like MIRAGE [30] which leverage the retrieval and LLM model internals for ensuring faithfulness of attributions. MIRAGE detects context-sensitive tokens in the LLM output and pairs them with retrieved documents, contributing to their prediction via saliency methods. `ir_explain` would be of immense utility in such attribution frameworks to detect salient terms through principled explainability approaches for debugging the retrieval model and the generative outputs. An overview of how `ir_explain` would aid in attribution for RAG systems is shown in Figure 3.

5 Summary and Future Work

This paper presents a Python library `ir_explain` implementing all the popular post-hoc approaches in ExIR. The library is integrated with Pyserini and `ir_datasets`, and integration with PyTerrier is in progress. Users of `ir_explain` can use several sparse and dense retrieval models and standard test collections for a rigorous analysis of ExIR methods. As we have demonstrated via a few use cases, this library would make it convenient for researchers to use, analyse and compare different explanation approaches, and identify the shortcomings of the current approaches. In turn, this should inspire and help future research in ExIR. As part of future work, we plan to integrate `ir_explain` with various visualization tools, and the ongoing probing components. Rigorous evaluation is another challenge for ExIR, with many studies providing mostly anecdotal evidence. We therefore plan to incorporate a rigorous evaluation framework for ExIR approaches within `ir_explain`.

Acknowledgement

We thank the anonymous reviewers for comments on earlier versions of this draft, which resulted in significant improvements in the current version. This work is partially supported by German Research Foundation (DFG), under the Project IREM with grant No. AN 996/1-1. Sourav Saha is supported by TCS Research Scholar program (Cycle 17). He also acknowledges Sobigdata++ team for a support to visit TU Delft.

References

[1] Enrique Amigó, Hui Fang, Stefano Mizzaro, and ChengXiang Zhai. 2017. Axiomatic Thinking for Information Retrieval: And Related Tasks. In *Proc. of SIGIR 2017*. 1419–1420.

[2] Avishek Anand, Lijun Lyu, Maximilian Idahl, Yumeng Wang, Jonas Wallat, and Zijian Zhang. 2022. Explainable Information Retrieval: A Survey. arXiv:2211.02405 [cs.IR]

[3] Avishek Anand, Procheta Sen, Sourav Saha, Manisha Verma, and Mandar Mitra. 2023. Explainable Information Retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 3448–3451. doi:10.1145/3539618.3594249

[4] Vijay Arya, Rachel K. E. Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Q. Vera Liao, Ronny Luss, Aleksandra Mojsilović, Sami Mourad, Pablo Pedemonte, Ramya Raghavendra, John Richards, Prasanna Sattigeri, Karthikeyan Shanmugam, Moninder Singh, Kush R. Varshney, Dennis Wei, and Yunfeng Zhang. 2021. AI Explainability 360 Toolkit. In *Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD)* (Bangalore, India) *(CODS-COMAD '21)*. Association for Computing Machinery, New York, NY, USA, 376–379. doi:10.1145/3430984.3430987

[5] Vijay Arya, Rachel K. E. Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Q. Vera Liao, Ronny Luss, Aleksandra MojsiloviÄ‡, Sami Mourad, Pablo Pedemonte, Ramya Raghavendra, John T. Richards, Prasanna Sattigeri, Karthikeyan Shanmugam, Moninder Singh, Kush R. Varshney, Dennis Wei, and Yunfeng Zhang. 2020. AI Explainability 360: An Extensible Toolkit for Understanding Data and Machine Learning Models. *Journal of Machine Learning Research* 21, 130 (2020), 1–6. http://jmlr.org/papers/v21/19-1035.html

[6] Bernd Bohnet, Vinh Q. Tran, Pat Verga, Roee Aharoni, Daniel Andor, Livio Baldini Soares, Massimiliano Ciaramita, Jacob Eisenstein, Kuzman Ganchev, Jonathan Herzig, Kai Hui, Tom Kwiatkowski, Ji Ma, Jianmo Ni, Lierni Sestorain Saralegui, Tal Schuster, William W. Cohen, Michael Collins, Dipanjan Das, Donald Metzler, Slav Petrov, and Kellie Webster. 2023. Attributed Question Answering: Evaluation and Modeling for Attributed Large Language Models. arXiv:2212.08037 [cs.CL] https://arxiv.org/abs/2212.08037

[7] Alexander Bondarenko, Maik Fröbe, Jan Heinrich Reimer, Benno Stein, Michael Völske, and Matthias Hagen. 2022. Axiomatic Retrieval Experimentation with ir_axioms. In *45th International ACM Conference on Research and Development in Information Retrieval (SIGIR 2022)*. ACM.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[9] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311 [cs.CL]

[10] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Term Weighting For First Stage Passage Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1533–1536. doi:10.1145/3397271.3401204

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

[12] Hui Fang, Tao Tao, and ChengXiang Zhai. 2004. A Formal Study of Information Retrieval Heuristics. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Sheffield, United Kingdom) *(SIGIR '04)*. Association for Computing Machinery, New York, NY, USA, 49–56. doi:10.1145/1008992.1009004

[13] Hui Fang and ChengXiang Zhai. 2005. An Exploration of Axiomatic Approaches to Information Retrieval. In *Proceedings of SIGIR 2005*. 480–487.

[14] Matthias Hagen, Michael Völske, Steve Göring, and Benno Stein. 2016. Axiomatic Result Re-Ranking. In *Proc. of CIKM 2016*. 721–730.

[15] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Trans. Inf. Syst.* 43, 2, Article 42 (Jan. 2025), 55 pages. doi:10.1145/3703155

[16] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. ACM, 39–48. doi:10.1145/3397271.3401075

[17] Jurek Leonhardt, Koustav Rudra, and Avishek Anand. 2023. Extractive Explanations for Interpretable Text Ranking. *ACM Trans. Inf. Syst.* 41, 4, Article 88 (March 2023), 31 pages. doi:10.1145/3576924

[18] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 9459–9474. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

[19] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. 2356–2362.

[20] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *CoRR* abs/2010.11386 (2020). arXiv:2010.11386 https://arxiv.org/abs/2010.11386

[21] Michael Llordes, Debasis Ganguly, Sumit Bhatia, and Chirag Agarwal. 2023. Explain Like I Am BM25: Interpreting a Dense Model's Ranked-List with a Sparse Approximation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Taipei, Taiwan) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 1976–1980. doi:10.1145/3539618.3591982

[22] Lijun Lyu and Avishek Anand. 2023. Listwise Explanations for Ranking Models Using Multiple Explainers. In *Advances in Information Retrieval*, Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (Eds.). Springer Nature Switzerland, Cham, 653–668.

[23] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 142–150. http://ai.stanford.edu/~amaas/data/sentiment/

[24] Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. 2021. Simplified Data Wrangling with ir_datasets. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2429–2436. doi:10.1145/3404835.3463254

[25] Craig Macdonald and Nicola Tonellotto. 2020. Declarative Experimentation in-Information Retrieval using PyTerrier. In *Proceedings of ICTIR 2020*.

[26] Iain Mackie, Jeffrey Dalton, and Andrew Yates. 2021. How Deep is your Learning: the DL-HARD Annotated Deep Learning Dataset. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[27] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 708–718. doi:10.18653/v1/2020.findings-emnlp.63

[28] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe.

2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL]

[29] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. arXiv:2101.05667 [cs.IR]

[30] Jirui Qi, Gabriele Sarti, Raquel Fernández, and Arianna Bisazza. 2024. Model Internals-based Answer Attribution for Trustworthy Retrieval-Augmented Generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 6037–6053. doi:10.18653/v1/2024.emnlp-main.347

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683 [cs.LG]

[32] Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and D. Reitter. 2021. Measuring Attribution in Natural Language Generation Models. *Computational Linguistics* 49 (2021), 777–840. https://api.semanticscholar.org/CorpusID:245502761

[33] Nils Reimers. 2021. Cross-Encoder for MS Marco. https://huggingface.co/cross-encoder/ms-marco-electra-base.

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

[35] Sourav Saha, Debapriyo Majumdar, and Mandar Mitra. 2022. Explainability of Text Processing and Retrieval Methods: A Critical Survey. arXiv:2212.07126 [cs.IR]

[36] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 3145–3153. http://proceedings.mlr.press/v70/shrikumar17a.html

[37] Jaspreet Singh and Avishek Anand. 2019. EXS: Explainable Search Using Local Model Agnostic Interpretability. In *Proc. of WSDM 2019*. 770–773.

[38] Jaspreet Singh and Avishek Anand. 2020. Model agnostic interpretability of rankers via intent modelling. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (Barcelona, Spain) *(FAT* '20)*. Association for Computing Machinery, New York, NY, USA, 618–628. doi:10.1145/3351095.3375234

[39] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *Proc of ICML 2017 (Proceedings of Machine Learning Research)*. 3319–3328.

[40] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL]

[41] Manisha Verma and Debasis Ganguly. 2019. LIRME: Locally Interpretable Ranking Model Explanation. In *Proc. of SIGIR 2019*. 1281–1284.

[42] Jonas Wallat, Hauke Hinrichs, and Avishek Anand. 2024. Causal Probing for Dual Encoders. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management* (Boise, ID, USA) *(CIKM '24)*. Association for Computing Machinery, New York, NY, USA, 2292–2303. doi:10.1145/3627673.3679556

[43] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.* 28, 4, Article 20 (nov 2010), 38 pages. https://doi.org/10.1145/1852102.1852106

[44] Dennis Wei, Sanjeeb Dash, Tian Gao, and Oktay Gunluk. 2019. Generalized linear rule models. In *International conference on machine learning*. PMLR, 6687–6696.

[45] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*. https://openreview.net/forum?id=zeFrfgyZln

[46] Fangyuan Xu, Yixiao Song, Mohit Iyyer, and Eunsol Choi. 2023. A Critical Evaluation of Evaluations for Long-form Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 3225–3245. doi:10.18653/v1/2023.acl-long.181

[47] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 1253–1256. doi:10.1145/3077136.3080721

[48] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf