

Constructing and Evaluating Declarative RAG Pipelines in PyTerrier

Craig Macdonald
University of Glasgow
Glasgow, United Kingdom

Jinyuan Fang
University of Glasgow
Glasgow, United Kingdom

Andrew Parry
University of Glasgow
Glasgow, United Kingdom

Zaiqiao Meng
University of Glasgow
Glasgow, United Kingdom

Abstract

Search engines often follow a pipeline architecture, where complex but effective reranking components are used to refine the results of an initial retrieval. Retrieval augmented generation (RAG) is an exciting application of the pipeline architecture, where the final component generates a coherent answer for the users from the retrieved documents. In this demo paper, we describe how such RAG pipelines can be formulated in the declarative PyTerrier architecture, and the advantages of doing so. Our PyTerrier-RAG extension for PyTerrier provides easy access to standard RAG datasets and evaluation measures, state-of-the-art LLM readers, and using PyTerrier's unique operator notation, easy-to-build pipelines. We demonstrate the succinctness of indexing and RAG pipelines on standard datasets (including Natural Questions) and how to build on the larger PyTerrier ecosystem with state-of-the-art sparse, learned-sparse, and dense retrievers, and other neural rankers.

CCS Concepts

- Information systems → Information retrieval.

Keywords

Retrieval Augmented Generation

ACM Reference Format:

Craig Macdonald, Jinyuan Fang, Andrew Parry, and Zaiqiao Meng. 2025. Constructing and Evaluating Declarative RAG Pipelines in PyTerrier. 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, 6 pages. <https://doi.org/10.1145/3726302.3730150>

1 Introduction

Information retrieval is increasingly concerned with complex architectures, beyond single-step sparse retrieval: for instance, learning-to-rank brought multi-stage architectures where candidate documents had extract features calculated, before being *re-ranked* by learned models [27]; more recently neural rerankers, such as those based on language models have become popular [25]; Similarly for dense retrievers, different pipeline stages can be conceived: query encoding, retrieval (including exact or approximate neighbourhood retrievers such as HNSW [33] or FAISS [17]), or even pseudo-relevance feedback [24, 48].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGIR '25, Padua, Italy

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1592-1/2025/07

<https://doi.org/10.1145/3726302.3730150>

PyTerrier¹ [31, 32] is a research platform that embraces multi-stage architectures for information retrieval research, by allowing indexing and retrieval pipelines to be formulated as expressions in Python: a rich ecosystem of transformations (indexers, retrievers, rerankers, features extractors) can be combined using intuitive operators into coherent pipelines. Each such transformation is applied on a common data model (i.e. data types or *relations* with expected attributes). The resulting pipelines are declarative in nature, in that the pipeline is formulated without considering the queries on which it will be applied - the resulting code is therefore declarative in nature and easily readable. Finally, pipelines can be evaluated and compared in a declarative experiment API, allowing the easy comparison of different retrieval systems for different query datasets, evaluation measures etc. Such evaluation avoids creating intermediate files (which are implementation by-products), as would have been required for traditional IR evaluation tools such as *trec_eval*.

On the other hand, while large language models (LLMs) have shown excellent memorisation able to often generate accurate answers to questions, they can hallucinate; this has given rise to Retrieval Augmented Generation (RAG) [23] as an increasingly popular multi-stage architecture, where the LLMs are provided the retrieved document as context in order to improve the quality of the generated answer. So-called *sequential* RAG models often employ a *retriever-reader* architecture, where a retriever first retrieves documents relevant to the query and a reader then generates an output based on the retrieved documents [12]. By grounding LLM outputs in retrieved documents, RAG models have demonstrated remarkable performance in a variety of tasks, such as question answering (QA) [26], fact checking [23] and dialogue generation [46]. However, sequential RAG models often struggle with tasks that require multi-step reasoning, such as multi-hop QA [45]. This limitation arises because single-step retrieval may fail to retrieve all the relevant information, leading to knowledge gaps in the reasoning process [40]. To address this issue, recent research has proposed *iterative* or adaptive RAG models, which perform multiple rounds of retrieval and reasoning to progressively gather the necessary information [1, 41, 45] and determine the answer [7, 9]. By dynamically retrieving additional information as needed, these approaches enhance RAG models' ability to handle complex reasoning tasks.

In particular, to benefit RAG researchers, we bring PyTerrier-RAG, which supports (i) QA datasets (e.g. Natural Questions, NQ [22]) - including providing pre-built indices, (ii) evaluation measures for QA (e.g. EM%, F1 [3, 50]), and access to a number of recent answer generation approaches. This is supplemented by standard classes allowing common styles of RAG approaches to be easily implemented, including iterative approaches characterised by IR-CoT [45]. Indeed, by implementing these in PyTerrier, RAG experimentation with other retrieval components is as easy as renaming one variable to another. Underlying all of this, an extended

¹  <https://github.com/terrier-org/pyterrier>

PyTerrier data model for RAG, and the intuitive PyTerrier operators allow pipelines to be constructed easily. PyTerrier-RAG’s source code and interactive Google Colab notebooks can be found at https://github.com/terrierteam/pyterrier_rag.

The remainder of this paper is structured as follows: We position PyTerrier-RAG with respect to other existing RAG frameworks in Section 2; In Section 3, we provide an overview of the necessary preliminaries of PyTerrier; Section 4 provides an overview of PyTerrier-RAG, describing advances in terms of data model, pipeline components, datasets and evaluation. We provide a full worked example in Section 5, and provide concluding remarks in Section 6.

2 Related Work

Among a number of commercial offerings, there are also several existing research-oriented RAG platforms that focus on different aspects of retrieval-augmented generation. For example, DSPy [20] is a framework designed for programming LLMs. It allows users to create compositional modules to build complex AI systems, including RAG pipelines and agentic applications [51]. However, DSPy does not offer pre-built RAG pipelines, requiring users to implement their own pipelines. In contrast, our framework offers both pre-processed datasets and pre-implemented RAG pipelines to enable succinct RAG experimentation. Recently, Rau et al. [38] introduced BERGEN, a benchmarking RAG system with a focus on question answering. BERGEN implements different retrievers and readers, enabling a systematic investigation of different components within a consistent setting. However, BERGEN is limited to the sequential RAG pipeline, consisting of a retriever followed by a reader. In contrast, our framework supports not only sequential pipelines but also more advanced RAG architectures, such as iterative RAG pipelines. Another platform, RAGChecker [39], is tailored to the evaluation of RAG systems. It offers an automatic evaluation framework designed to assess each component of the RAG pipelines.

In addition, among the existing RAG research platforms, FlashRAG [16] is the most similar to our framework. It provides a comprehensive environment for developing and evaluating RAG pipelines, offering pre-built datasets, retrieval models, and different readers. However, there are key differences between the two platforms. One key difference is that FlashRAG relies on configuration files for setup and management, whereas our framework eliminates the need for such files. This design choice in PyTerrier-RAG allows users to configure and modify their RAG pipelines directly within their code without modifying external configuration files - ideal for notebook-based agile research. Additionally, PyTerrier-RAG adopts a declarative style, allowing users to easily compose and modify different RAG pipelines in a plug-and-play manner, thereby enhancing flexibility and usability. Another key difference is that FlashRAG offers a limited set of retrievers and rerankers. For example, it does not support multi-vector dense retrieval models such as ColBERT [21] or learned sparse retrieval models such as SPLADE [11], which are two families of strong retrieval models from the literature. In contrast, our framework is built upon the PyTerrier platform, enabling access to a large ecosystem of retrievers and rerankers.

3 PyTerrier Preliminaries

PyTerrier operates on relations with known primary keys and optional attributes. For instance, queries are represented by the Q type with schema $Q(qid, query, \dots)$; and documents by the D type with

Table 1: Exemplar PyTerrier operators for combining transformers.

Op.	Name	Description
>>	<i>then</i>	Pass the output from one transformer to the next transformer
+	<i>linear combine</i>	Sum the query-document scores of the two retrieved results lists
	<i>set union</i>	Make the set union of documents from the two retrieved results lists
%	<i>rank cutoff</i>	Shorten a retrieved results list to the first K elements

schema $D(docno, text, \dots)$. Retrieved documents, R , are defined as a relationship type between Q and D , i.e. $R \subset Q \times D$, with schema $R(qid, docno, score, rank)$; similarly relevant assessments are also a subset of $Q \times D$, i.e. $RA(qid, docno, label)$. All these relations can be instantiated in Python as Pandas Dataframes, or more simply iterable lists of dictionaries with the required and optional attributes.

Next, each component of an indexing or retrieval pipelines forms a transformation from one relation type to one or another type. We can characterise different families of transformation that are applied in indexing & retrieval:

- *Retrieval*: $Q \rightarrow R$
- *Reranking*: $R \rightarrow R$
- *Query rewriting*: $Q \rightarrow Q$
- *Document expansion*: $D \rightarrow D$
- *Pseudo-relevance feedback*: $R \rightarrow Q$
- *Indexing*: $D \rightarrow \emptyset$ (a terminal operation)

For instance, one can create a retriever on a given index, which when provided with a set of one or more queries, returns 0 or more retrieved documents for each query. We call such components *transformers* as they transform from one datatype to another.

Typically, indexing or retrieval pipelines are combinations of several such transformer components. An imperative use would involve getting results from one transformer before inputting them into another. Instead, PyTerrier suggests a declarative manner, in which pipelines are constructed before applying them to unseen input data.² To aid in this, several operators are defined on transformers (implemented through the use of operator overloading within Python). For example, a hybrid retriever that linearly combines the scores of two different retrievers can be expressed as $ret1 + ret2$. The result of such an expression is itself a transformer. The most frequently used operator is \gg (also called ‘then’), and analogous to the $|$ operator in Unix shells, is defined as follows $a \gg b$: a is applied on its input, and the output of a is then passed as input to b . Salient PyTerrier operators and their meanings are summarised in Table 1. Each operator has relational algebra semantics, as explored in [31].

A key advantage of the PyTerrier ecosystem is the richness of its ecosystem of plugins - instead of bloating the core platform with additional functionality, our preference is that we, and others, release plugins that can be maintained separately, taking advantage of the decomposable of indexing and retrieval pipelines and the extensible data model. For instance, a dense retrieval pipeline may be expressed as $query_encoder \gg retriever$, with the Q -typed output of the $query_encoder$ being augmented with an additional

² Advantages include, for instance, deterring bad practices such as excessive preprocessing of queries that makes testing on unseen queries difficult to achieve.

query_emb column. Table 2 provides an overview of many PyTerrier plugins, and gives sample indexing and/or retrieval pipelines.

The rich PyTerrier ecosystem demonstrates the variety of different retrieval pipelines that can be used for retrieval augmented generation. In the next section, we highlight the needed additions to support experimentation within retrieval augmented generation.

4 PyTerrier-RAG

To better support RAG, and allow researchers working on RAG full access to the PyTerrier retrieval and reranking ecosystem, PyTerrier-RAG makes several new additions: (i) an extended data model (Section 4.1); (ii) a suite of common answer generators (often called *readers*, Section 4.2), and ways to translate retrieval output into context for a language-model based reader; (iii) support for iterative RAG architectures [45]; (iv) access to standard QA datasets (Section 4.4); and evaluation measures (Section 4.5).

4.1 Data Model Extensions

PyTerrier’s primary datatypes, as outlined in Section 3, include Q , a set of queries/questions, and R , a set of retrieved documents. Building on these, we add datatypes for standard RAG QA experiments: answers for a question/query, $A(qid, qanswer)$; gold answers $GA(qid, [ganswer])$; and, extending Q , Qc which contains context, i.e. $Qc(qid, query, qcontext)$. Using these datatypes, we can define new classes of transformations between datatypes:

- 0-shot answer generation: $(Q \rightarrow A)$.
- Creation of RAG Context: $R \rightarrow Qc$ - formulates context from a retriever.
- Reader: $Qc \rightarrow A$ - takes retrieved documents as context to an LLM to generate an answer.

We next describe the readers implemented within PyTerrier-RAG.

4.2 New Reader Components

The primary novel component of this framework is the Reader transformer ($Qc \rightarrow A$). For a given input text, a reader provides a single answer. To allow flexibility in models, we implement both locally served and API-based language models under a backend class. A Backend implementation need only implement a `.generate()` function that accepts an iterable of prompt strings and returns an iterable of answer strings; overall, data structure processing is handled by the Reader - e.g. formulation as A datatype.

```
1 causal_backend = HuggingFaceBackend("meta-llama/Llama-3.1-8B")
2 seq2seq_backend = Seq2SeqLMBackend("google/flan-t5-base")
3 openai_backend = OpenAIBackend("gpt-4o-mini")
```

Though a RAG pipeline is generally considered to be a retrieval system followed by a generative component, we further abstract the generative component to align with the declarative nature of PyTerrier. To allow flexible setups in terms of retrieval and steps for context creation, the Reader class expects only a Qc datatype, with a separate component handling the collation of intermediate context. Retrieval output is aggregated by a separate Concatenator class, i.e $R \rightarrow Qc$. So a sequential RAG pipeline is as follows:

```
1 openai_reader = Reader(backend=openai_backend)
2 bm25 = pt.Artifact.from_hf('pyterrier/ragwiki-terrier').bm25()
3 bm25_monot5 = bm25 >> MonoT5()
4 pipeline = bm25_monot5 >> Concatenator() >> openai_reader
```

Furthermore, within a reader, the prompt can be a string instruction or a `PromptTransformer` ($Qc \rightarrow Qc$) to facilitate more complex instructions. The latter receives a Qc dataframe with the question and context aggregated from a retriever. Internally, a prompt template format creates a single string from question and context to be passed to a Backend within the reader.

Finally, we also support Fusion-in-Decoder (FiDReader) architectures [15]; however, due to the separate encoding of documents and fine-tuned nature of this architecture, this class expects a ranking (R) as opposed to a combined context Qc .

4.3 Non-Sequential RAG Pipelines

Beyond sequential RAG, several frameworks have been proposed that apply an iterative process where the output of the reader component forms part of the input of the retriever over multiple cycles. We exemplify this paradigm with an implementation of a popular iterative method, IRCoT [45], which, after each iteration, checks the reader’s output against a user-provided exit condition (for example, checking for the presence of a stopping phrase specified by a prompt such as “so the answer is”) as shown below:

```
1 exit_condition = lambda x: "so_the_answer_is" in x.qanswer.iloc[0]
2 ircot_bm25_monot5_openai = IRCOT(retriever=bm25 >> monoT5,
3                                     backend=openai_backend,
4                                     exit_condition=exit_condition)
```

The ability to refine the retriever, e.g., by adding a monoT5 cross-encoder, is easily viewable. More recent iterative works apply reasoning on entities and relationships extracted as knowledge graphs - for instance, our own recent works in this area, REANO [8] and TRACE [10], can be expressed as PyTerrier-RAG pipelines. For example, in TRACE, documents must have entities and corresponding knowledge relationship triples identified (offline or online, stored as extra columns in the R datatype); an LLM is prompted iteratively to build a reasoning chain with knowledge triples extracted from retrieved documents to answer a question.

4.4 Datasets and Corpora

To facilitate succinct research with RAG pipelines, PyTerrier-RAG offers ten pre-processed benchmark datasets, covering a variety of datasets commonly used in RAG studies [12]. Table 3 provides a summary of these datasets, including their respective tasks, corpora and statistics. Specifically, our framework provides programmatic access to datasets for various retrieval-augmented tasks, including general domain QA, multi-hop QA, dialogue generation, and fact-checking. All datasets have been processed to be compatible with PyTerrier pipelines, including three key attributes: `qid`, `query` and `answer`. Moreover, our framework offers flexible dataset integration, allowing users to seamlessly incorporate datasets from other platforms, such as FlashRAG [16]. Indexing new datasets in PyTerrier is easy - they need only be expressed as iterable dictionaries, easily obtainable from JSON files.

In addition to the benchmark datasets, our framework offers multiple retrieval corpora, such as Wikipedia, HotPotQA, 2Wiki and MuSiQue. Specifically, the Wikipedia corpus is based on the widely used December 20, 2018 Wikipedia dump released by DPR [19], containing about 21M passages. The HotPotQA corpus, originally introduced alongside the HotPotQA dataset, consists of around 5 million documents. The 2Wiki and MuSiQue corpora are constructed from

Table 2: Examples of PyTerrier plugins, with links to the corresponding GitHub repositories and sample pipelines.

Plugin	Functionality	Sample Pipeline
pyterrier-anserini	Indexing & Retrieval using Anserini [49]	AnseriniIndex.from_hf('macavaney/msmarco-passage.anserini').bm25()
pyterrier-pisa	Fast Indexing & Retrieval using PISA [29, 35]	PisaIndex("./cord19-pisa").bm25()
pyterrier-t5	monoT5 & duoT5 reranking [36]	bm25 >> MonoT5() % 10 >> DuoT5()
pyterrier-splade	SPLADE indexing and retrieval	splade_ret = splade >> pt.terrier.Retriever(index, wmodel="Tf")
pyterrier-doc2query	Doc2Query(–) [13, 36] document expansion	index_pipe = Doc2Query() >> pt.terrier.IterDictIndexer()
pyterrier-genrank	Listwise generative rerankers [5]	bm25 >> LLMReranker("castorini/rank_vicuna_7b_v1")
pyterrier-dr	Dense Retrieval	TasB() >> FlexIndex("./cord19")
pyterrier-colbert	ColBERT [21] Dense Retrieval	ColBERTFactory().end_to_end()

Table 3: Summary of datasets and corpus in PyTerrier-RAG, where “2Wiki” denotes “2WikiMultihopQA”.

Task	Dataset	Corpus	# Train	# Dev	# Test
QA	Natural Questions [22]	Wikipedia	79,168	8,757	3,610
	TriviaQA [18]	Wikipedia	78,785	8,837	11,313
	WebQuestions [2]	Wikipedia	3,778	—	2,032
	PopQA [34]	Wikipedia	—	—	14,267
Multi-Hop QA	HotpotQA [50]	HotpotQA	90,447	7,405	—
	MuSiQue [44]	MuSiQue	19,938	2,417	—
	2Wiki [14]	2Wiki	15,000	12,576	—
	Bamboogle [37]	Wikipedia	—	—	125
Dialogue Generation	WoW [6]	Wikipedia	63,734	3,054	—
Fact Checking	FEVER [42]	Wikipedia	104,966	10,444	—

their respective contexts following the procedure outlined in [45], and containing about 431K and 117K documents, respectively.

We also provide some pre-built indices for common corpora available via HuggingFace datasets, and accessed using a single line of Python code (see the first example snippet in Section 5)³.

4.5 Evaluation

Evaluation for QA in PyTerrier follows the declarative nature in the platform: A `pt.Experiment` is called with four essential arguments: (1) the systems being compared; (2) the queries being used for evaluation; (3) the ground truth; and (4) the evaluation measures to be calculated. QA datasets differ from classical IR datasets in that the success is typically measured in terms of the similarity to one or more accepted gold-truth answers. Hence for evaluation in PyTerrier-RAG, we use the classical `pt.Experiment()` function from PyTerrier, but change (i) the type of the ground truth from *RA* (document-level relevance assessments) to *GA* (gold answers); and (ii) provide definitions for the standard textual overlap measures: *Exact Match* [3]; *F1* [50] (both based on the implementations provided in the DPR repository [19]); and ROUGE measures. An example `pt.Experiment()` is shown in Section 5.

Another form of evaluation is the use of language models for evaluating generated answers. We provide BERTScore [52], which compares generated answers to known relevant documents [43], as well as prompts for LLMs to provide ratings on answers [53].

`pt.Experiment()` also offers other advantages such as significance testing, multiple-testing correction, ability to change the batch size. One notable recent improvement is prefix-computation [30]. This examines the pipelines of the systems being evaluated, identifies any common prefix of the constituent pipeline components, and applies that prefix to the input topics only once. The results on the prefix can be reused for the other pipelines. For example, if an

³ More details of such artefacts are provided in [28]

experiment was evaluating the impact on sequential RAG answer quality of changing the number of retrieved documents from BM25, precomputation would compute the BM25 results only once.

5 Worked Example

To simplify experimentation and facilitate efficient retrieval, our framework provides a variety of pre-built indices for common datasets, including those based on BM25 and the E5 [47] dense retrieval model. These indices allow users to quickly set up retrieval pipelines without the need for extensive preprocessing or index construction. Below, we show an example experiment evaluating answer quality when comparing using the top 10 BM25 retrieved documents versus the top 10 from E5. The output is a table of F1 and EM measurements for both pipelines – example output is shown in the provided live notebooks.

```

1 dataset = pt.get_dataset('rag:nq')
2 sparse_index = pt.Artifact.from_hf('pyterrier/ragwiki-terrier')
3 e5_emb_index = pt.Artifact.from_hf('pyterrier/ragwiki-e5.flex')
4 bm25 = sparse_index.bm25(include_fields=['docno', 'title', 'text'])
5 e5 = E5() >> e5_emb_index.retriever()
6 raw_text = pt.text.get_text(sparse_index, ['title', 'text'])
7 fid = pyterrier_rag.readers.T5FiD("terrierteam/t5fid_base_nq")
8 bm25_fid = bm25 %
9 e5_fid = e5 %
10 pt.Experiment(
11     [bm25_fid, e5_fid],
12     dataset.get_topics('dev'),
13     dataset.get_answers('dev'),
14     [pyterrier_rag.measures.F1, pyterrier_rag.measures.EM])

```

6 Conclusions

This demonstration paper aims to summarise recent developments in a RAG plugin for PyTerrier, showing how the PyTerrier data-model can be extended to RAG use cases, and allows state-of-the-art retrieval techniques to be easily interchanged to examine their impact upon answer quality. We believe that easily expressing different retrieval pipelines is of particular importance in RAG research, as the nature of LLMs is that consumption of information is lossy, and differ from previous expectations in the IR about ordering documents by descending likelihood of estimated relevance [4, 43]. In future work, we will continue to supplement PyTerrier-RAG with more datasets and more reader implementations.

Acknowledgments

We thank the authors of FlashRAG [16] for their efforts in standardising the various QA datasets into a single HuggingFace repository. We thank our colleagues Sean MacAvaney for input in PyTerrier-RAG, and Fangzheng Tian for the BERTScore implementation.

References

- [1] Akari Asai, Zequi Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. <https://openreview.net/forum?id=hSyW5go0v8>
- [2] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 1533–1544. <https://aclanthology.org/D13-1160/>
- [3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, 1870–1879. <https://doi.org/10.18653/V1/P17-1171>
- [4] Florin Cuconas, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonello, and Fabrizio Silvestri. 2024. The Power of Noise: Redefining Retrieval for RAG Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (Eds.). ACM, 719–729. <https://doi.org/10.1145/3626772.3657834>
- [5] Kaustubh D. Dhole. 2024. PyTerrier-GenRank: The PyTerrier Plugin for Reranking with Large Language Models. *CoRR* abs/2412.05339 (2024). <https://doi.org/10.48550/arXiv.2412.05339> arXiv:2412.05339
- [6] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of Wikipedia: Knowledge-Powered Conversational Agents. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. <https://openreview.net/forum?id=r1l73iRqKm>
- [7] Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. 2024. REANO: Optimising Retrieval-Augmented Reader Models through Knowledge Graph Generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, 2094–2112. <https://doi.org/10.18653/V1/2024.ACL-LONG.115>
- [8] Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. 2024. REANO: Optimising Retrieval-Augmented Reader Models through Knowledge Graph Generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, 2094–2112. <https://doi.org/10.18653/V1/2024.ACL-LONG.115>
- [9] Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. 2024. TRACE the Evidence: Constructing Knowledge-Grounded Reasoning Chains for Retrieval-Augmented Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 8472–8494. <https://aclanthology.org/2024.findings-emnlp.496>
- [10] Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. 2024. TRACE the Evidence: Constructing Knowledge-Grounded Reasoning Chains for Retrieval-Augmented Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 8472–8494. <https://aclanthology.org/2024.findings-emnlp.496>
- [11] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 2288–2292. <https://doi.org/10.1145/3404835.3463098>
- [12] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *CoRR* abs/2312.10997 (2023). <https://doi.org/10.48550/arXiv.2312.10997> arXiv:2312.10997
- [13] Mitko Gospodinov, Sean MacAvaney, and Craig Macdonald. 2023. Doc2Query-: When Less is More. In *Advances in Information Retrieval - 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2-6, 2023, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 13981)*, Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo (Eds.). Springer, 414–422. https://doi.org/10.1007/978-3-031-28238-6_31
- [14] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, Donia Scott, Núria Bel, and Chengqing Zong (Eds.). International Committee on Computational Linguistics, 6609–6625. <https://doi.org/10.18653/V1/2020.COLING-MAIN.580>
- [15] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty (Eds.). Association for Computational Linguistics, 874–880. <https://doi.org/10.18653/V1/2021.EACL-MAIN.74>
- [16] Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research. *CoRR* abs/2405.13576 (2024). <https://doi.org/10.48550/arXiv.2405.13576> arXiv:2405.13576
- [17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data* 7, 3 (2021), 535–547. <https://doi.org/10.1109/TBDA.2019.2921572>
- [18] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, 1601–1611. <https://doi.org/10.18653/V1/P17-1147>
- [19] Vladimir Karpukhin, Barlas Ouzi, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 6769–6781. <https://doi.org/10.18653/V1/2020.EMNLP-MAIN.550>
- [20] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. *CoRR* abs/2310.03714 (2023). <https://doi.org/10.48550/ARXIV.2310.03714> arXiv:2310.03714
- [21] 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*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 39–48. <https://doi.org/10.1145/3397271.3401075>
- [22] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: a Benchmark for Question Answering Research. *Trans. Assoc. Comput. Linguistics* 7 (2019), 452–466. https://doi.org/10.1162/TACL_A_00276
- [23] 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 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>
- [24] Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. 2021. Pseudo Relevance Feedback with Deep Language Models and Dense Retrievers: Successes and Pitfalls. *CoRR* abs/2108.11044 (2021). arXiv:2108.11044 <https://arxiv.org/abs/2108.11044>
- [25] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Morgan & Claypool Publishers. <https://doi.org/10.2200/S01123ED1V01Y202108HLT053>
- [26] Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvassy, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. RA-DIT: Retrieval-Augmented Dual Instruction Tuning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. <https://openreview.net/forum?id=22OTbutug9>
- [27] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.
- [28] Sean MacAvaney. 2025. Artifact Sharing for Information Retrieval Research. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. <https://doi.org/10.1145/3726302.3730147>

- [29] Sean MacAvaney and Craig Macdonald. 2022. A Python Interface to PISA!. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai (Eds.). ACM, 3339–3344. <https://doi.org/10.1145/3477495.3531656>
- [30] Sean MacAvaney and Craig Macdonald. 2025. On Precomputation and Caching in Information Retrieval Experiments with Pipeline Architectures. In *Proceedings of WOWS workshop at ECIR 2025*. <https://doi.org/10.48550/arXiv.2504.09984>
- [31] Craig Macdonald and Nicola Tonellootto. 2020. Declarative Experimentation in Information Retrieval using PyTerrier. In *ICTIR '20: The 2020 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Norway, September 14-17, 2020*, Krisztian Balog, Vinay Setty, Christina Lioma, Yiqun Liu, Min Zhang, and Klaus Berberich (Eds.). ACM, 161–168. <https://doi.org/10.1145/3409256.3409829>
- [32] Craig Macdonald, Nicola Tonellootto, Sean MacAvaney, and Iadh Ounis. 2021. PyTerrier: Declarative Experimentation in Python from BM25 to Dense Retrieval. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*. ACM, 4526–4533. <https://doi.org/10.1145/3459637.3482013>
- [33] Yury A. Malkov and Dmitry A. Yashunin. 2016. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *CoRR* abs/1603.09320 (2016). arXiv:1603.09320 <http://arxiv.org/abs/1603.09320>
- [34] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hanneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 9802–9822. <https://doi.org/10.18653/V1/2023.ACL-LONG.546>
- [35] Antonio Mallia, Michal Siedlaczek, Joel M. Mackenzie, and Torsten Suel. 2019. PISA: Performant Indexes and Search for Academia. In *Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris, France, July 25, 2019 (CEUR Workshop Proceedings, Vol. 2409)*, Ryan Clancy, Nicola Ferro, Claudia Hauff, Jimmy Lin, Tetsuya Sakai, and Ze Zhong Wu (Eds.). CEUR-WS.org, 50–56. <https://ceur-ws.org/Vol-2409/docker08.pdf>
- [36] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. *CoRR* abs/2101.05667 (2021). arXiv:2101.05667 <https://arxiv.org/abs/2101.05667>
- [37] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and Narrowing the Compositional Gap in Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 5687–5711. <https://doi.org/10.18653/V1/2023.FINDINGS-EMNLP.378>
- [38] David Rau, Hervé Déjean, Nadezhda Chirkova, Thibault Formal, Shuai Wang, Stéphane Clinchant, and Vassilina Nikouline. 2024. BERGEN: A Benchmarking Library for Retrieval-Augmented Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 7640–7663. <https://aclanthology.org/2024.findings-emnlp.449>
- [39] Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. RAGChecker: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, Amir Globerson, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (Eds.). http://papers.nips.cc/paper/_files/paper/2024/hash/27245589131d17368cccd990cbf16e-Abstract-Datasets_and_Benchmarks_Track.html
- [40] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 9248–9274. <https://doi.org/10.18653/V1/2023.FINDINGS-EMNLP.620>
- [41] Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. DRAGIN: Dynamic Retrieval Augmented Generation based on the Real-time Information Needs of Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikanth (Eds.). Association for Computational Linguistics, 12991–13013. <https://doi.org/10.18653/V1/2024.ACL-LONG.702>
- [42] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, Marilyn A. Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, 809–819. <https://doi.org/10.18653/V1/N18-1074>
- [43] Fangzheng Tian, Debasis Ganguly, and Craig Macdonald. 2025. Is Relevance Propagated from Retriever to Generator in RAG?. In *Advances in Information Retrieval - 47th European Conference on Information Retrieval, ECIIR 2025, Lucca, Italy, April 6-10, 2025, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 15572)*, Claudia Hauff, Craig Macdonald, Dietmar Jannach, Gabriella Kazai, Franco Maria Nardini, Fabio Pinelli, Fabrizio Silvestri, and Nicola Tonellootto (Eds.). Springer, 32–48. https://doi.org/10.1007/978-3-031-88708-6_3
- [44] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop Questions via Single-hop Question Composition. *Trans. Assoc. Comput. Linguistics* 10 (2022), 539–554. https://doi.org/10.1162/TACL_A_00475
- [45] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 10014–10037. <https://doi.org/10.18653/V1/2023.ACL-LONG.557>
- [46] Hongru Wang, Wenyu Huang, Yang Deng, Rui Wang, Zezhong Wang, Yufei Wang, Fei Mi, Jeff Z. Pan, and Kam-Fai Wong. 2024. UniMS-RAG: A Unified Multi-source Retrieval-Augmented Generation for Personalized Dialogue Systems. *CoRR* abs/2401.13256 (2024). <https://doi.org/10.48550/ARXIV.2401.13256> arXiv:2401.13256
- [47] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Dixin Jiang, Rangan Majumder, and Furu Wei. 2022. Text Embeddings by Weakly-Supervised Contrastive Pre-training. *CoRR* abs/2212.03533 (2022). <https://doi.org/10.48550/ARXIV.2212.03533> arXiv:2212.03533
- [48] Xiao Wang, Craig Macdonald, Nicola Tonellootto, and Iadh Ounis. 2021. Pseudo-Relevance Feedback for Multiple Representation Dense Retrieval. In *ICTIR '21: The 2021 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Canada, July 11, 2021*, Faegheh Hasibi, Yi Fang, and Akiko Aizawa (Eds.). ACM, 297–306. <https://doi.org/10.1145/3471158.3472250>
- [49] 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, August 7-11, 2017*, Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryon W. White (Eds.). ACM, 1253–1256. <https://doi.org/10.1145/3077136.3080721>
- [50] Zhihlin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, 2369–2380. <https://doi.org/10.18653/V1/D18-1259>
- [51] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izahk Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/forum?id=WE_yluYUL-X
- [52] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. <https://openreview.net/forum?id=SkeHuCVFDr>
- [53] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper/_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html