Rankify: A Comprehensive Python Toolkit for Retrieval, Re-Ranking, and Retrieval-Augmented Generation

Abdelrahman Abdallah[1], **Bhawna Piryani**[1], **Jamshid Mozafari**[1]
Andreas Herzinger[1], **Jamie Holdcroft**[2*], **Adam Jatowt**[1]
[1]University of Innsbruck
[2]University of New South Wales
{abdelrahman.abdallah,adam.jatowt}@uibk.ac.at

Abstract

Building retrieval-augmented generation (RAG) systems often requires combining separate tools for retrieval, re-ranking, and generation, with incompatible data formats, evaluation pipelines, and deployment workflows. We present RANKIFY, an open-source Python toolkit that unifies these stages in a single modular framework[1,2,3,4]. RANKIFY provides 42 benchmark datasets with pre-retrieved documents and pre-built indices, 15 retrievers (sparse, dense, and reasoning-augmented), and 24 re-ranking models spanning 41 pointwise, pairwise, and listwise variants. It also supports **6** RAG strategies across 4 inference backends (Hugging Face, vLLM, LiteLLM, and OpenAI), enabling consistent experimentation from local models to hosted APIs. A unified pipeline interface allows users to compose retrieve–rerank–generate workflows in a few lines of code, while an agentic assistant (`RankifyAgent`), a REST server (`RankifyServer`), and an interactive web playground support deployment and non-programmatic exploration. The framework allows extensive experimentation. We showcase it exploring 200+ configurations on QA and BEIR/TREC benchmarks with six generator LLMs, which demonstrate that re-ranking consistently improves downstream performance, yielding gains of 5–15 points in Exact Match and up to 8.5 points in RAGAS context precision across diverse retriever–generator combinations. RANKIFY is released under the Apache-2.0 License.

1 Introduction

Retrieval-augmented generation (RAG) and retrieval-based question answering have become standard patterns for grounding large language model outputs in external knowledge (Gao et al., 2023; Izacard and Grave, 2020; Zhu et al., 2021). In these systems, the quality of the final answer depends critically on the quality of the retrieved and selected evidence. Modern pipelines therefore adopt a two-stage design: a fast retriever proposes candidate documents, and a more precise re-ranker refines the candidate list to maximize relevance (Yates et al., 2021; Nogueira et al., 2019; Ren et al., 2021). This retrieve-then-rerank paradigm has proved effective across a wide range of benchmarks and downstream applications.

The main barriers are **fragmentation** and **inconsistency**. IR toolkits such as Pyserini (Lin et al., 2021) provide strong retrieval infrastructure; re-ranking libraries like Rerankers (Clavié, 2024) ease cross-encoder integration; and RAG frameworks such as FlashRAG (Jin et al., 2024) and AutoRAG (Kim et al., 2024) offer generation-oriented workflows. However, connecting these components into a unified, reproducible pipeline—with consistent data structures, batching, and evaluation—still demands substantial glue code. Orchestration frameworks (LangChain (Chase, 2022), DSPy (Khattab et al., 2023)) simplify composition but are not designed as benchmarking environments for systematically comparing retrieval and ranking components across many datasets.

We present RANKIFY, an open-source Python toolkit and live demonstration system that addresses these gaps. RANKIFY provides:

1. **Unified API:** A modular `pipeline` interface that standardizes retrieval, re-ranking, and RAG execution so that components can be swapped without rewriting glue code.

2. **Broad coverage:** 42 datasets (each with 1 000 pre-retrieved documents per query) with pre-built indices, 15 retrievers (sparse, dense, reasoning-augmented), 24 re-ranking models

*The work has been done at the University of Innsbruck.
[1]PyPI: https://pypi.org/project/rankify/
[2]GitHub: https://github.com/DataScienceUIBK/Rankify
[3]Docs: https://rankify.readthedocs.io
[4]Video: https://youtu.be/kkLzomrM2ec

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 208–219

(41 variants), and **six** RAG strategies across four inference backends.

3. **Custom indexing:** `rankify-index` builds FAISS indices for arbitrary corpora using local or API-based embeddings, enabling evaluation beyond the included benchmarks.

4. **Comprehensive evaluation:** Built-in metrics for retrieval (Top-k, nDCG, MAP, MRR), re-ranking, and generation quality (EM, F1, RAGAS-inspired faithfulness/relevancy/precision (Es et al., 2024)).

5. **Agentic assistance and deployment:** `RankifyAgent` recommends pipeline configurations under task and hardware constraints; `RankifyServer` exposes any pipeline as a REST API; and an interactive web playground enables non-programmatic exploration.

Table 1 compares RANKIFY with existing toolkits. RANKIFY is the only toolkit providing pre-retrieved datasets and pre-built indices for 42 benchmarks, 24 re-ranking models spanning all three paradigms, and end-to-end RAG evaluation with RAGAS metrics in a single `pip`-installable package. RANKIFY targets NLP researchers and practitioners who need reproducible, end-to-end retrieval and RAG experimentation without building custom infrastructure.

2 Related Work

Information retrieval has progressed from lexical matching (BM25 (Robertson et al., 1995)) to dense retrievers that learn semantic matching via neural encoders (DPR (Yates et al., 2021), ANCE (Xiong et al., 2021), ColBERT (Khattab and Zaharia, 2020)). Re-ranking methods further refine candidate sets, ranging from pointwise cross-encoders (Nogueira and Cho, 2019) to listwise LLM-based rankers such as RankGPT (Sun et al., 2023) and RankT5 (Zhuang et al., 2023). RAG integrates retrieved evidence into LLM prompting for knowledge-intensive tasks (Lewis et al., 2020; Izacard and Grave, 2020). Several toolkits support parts of this pipeline: Pyserini (Lin et al., 2021) provides reproducible retrieval, Rerankers (Clavié, 2024) targets re-ranking, and FlashRAG (Jin et al., 2024) / AutoRAG (Kim et al., 2024) offer RAG workflows; LangChain (Chase, 2022) and DSPy (Khattab et al., 2023) simplify composition. However, these systems typically trade off breadth of retriever–reranker coverage,

3 The RANKIFY Toolkit

RANKIFY is a modular Python library built on PyTorch (Paszke et al., 2019) and Pyserini (Lin et al., 2021). It is installable in one command:

```
$ pip install rankify
```

The architecture comprises five modules: **Datasets**, **Retrievers**, **Re-Rankers**, **RAG Generators**, and **Evaluation**. These modules operate independently yet share a common data model (`Document`, `Question`, `Answer`, `Context`), enabling seamless composition. An example is shown in Figure 1.

```python
from rankify import pipeline

# Retrieve + rerank in one line
rerank = pipeline(
  "rerank",
  retriever="bge",
  reranker="flashrank",
  index_type="wiki",
  n_docs=10,
)
docs = rerank("What is machine learning?")
print(docs[0].text[:200])
```

Figure 1: Using RANKIFY: constructing and running a retrieve + rerank pipeline with the `pipeline()` API;

3.1 Datasets and Pre-built Indices

RANKIFY provides a standardized schema for **42 datasets** across 12 task categories: open-domain QA (NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), SQuAD (Rajpurkar et al., 2016), WebQ (Berant et al., 2013), PopQA (Mallen et al., 2022), and 12 others), multi-hop QA (Ho et al., 2020; Yang et al., 2018; Trivedi et al., 2022; Press et al., 2023), temporal QA (Wang et al., 2022a; Piryani et al., 2024), long-form QA (Fan et al., 2019; Stelmakh et al., 2022), multiple-choice (Hendrycks et al., 2021; Clark et al., 2018; Lin et al., 2022), entity linking (Tedeschi et al., 2021; Sciavolino et al., 2021), slot filling (Levy et al., 2017; ElSahar et al., 2018), dialogue (Dinan et al., 2019), fact verification (Thorne et al., 2018), summarization (Hayashi et al., 2020), and IR benchmarks (BEIR (Thakur et al., 2021), DL19/20 (Craswell et al., 2020)). Complete dataset statistics are provided in Appendix C.

Each dataset ships with 1 000 pre-retrieved documents per query (processed by BM25, DPR, and ColBERT from Wikipedia or MS MARCO) and pre-built FAISS indices, eliminating the costly indexing step. Two large-scale corpora are provided out of the box: **Wikipedia** (21 M passages,

Feature	FlashRAG	AutoRAG	FastRAG	Rerankers	RANKIFY
Dataset Support					
# Datasets	38	4	0	0	**42**
Pre-retrieved docs	✗	✗	✗	✗	✓
Pre-built indices	✗	✗	✗	✗	✓
Online retrieval	✗	✗	✗	✗	✓
Model Support					
# Retrievers	3	2	1	0	**15**
# Re-rankers (methods)	2	7	3	15	**24 (41)**
Pointwise re-ranking	Limited	✓	✓	✓	✓
Pairwise re-ranking	✗	Limited	✗	Limited	✓
Listwise re-ranking	✗	✗	✗	Limited	✓
# RAG methods	16	1	7	0	6
Evaluation Capabilities					
Retrieval metrics	✓	✓	✗	✓	✓
Re-ranking metrics	Limited	✓	✗	✓	✓
RAG metrics (EM, F1)	✓	✓	✗	✗	✓
RAGAS-inspired metrics	✗	✗	✗	✗	✓
Full pipeline evaluation	Limited	Limited	✗	✗	✓
Design Philosophy					
Training support	✗	✗	✗	✗	✗
Benchmarking focus	✓	Limited	✗	✓	✓
Reproducibility	Limited	Limited	Limited	✓	✓
Modularity	✓	✓	✓	✓	✓
Documentation	✓	✓	Limited	✗	✓

Table 1: Comparison of RANKIFY with existing IR and RAG toolkits. RANKIFY uniquely combines 24 re-ranking models with 41 methods (pointwise/pairwise/listwise), online retrieval, pre-retrieved datasets with pre-built indices, and RAGAS evaluation metrics.

Dec. 2018 dump (Karpukhin et al., 2020)) and **MS MARCO** (Nguyen et al., 2017), each indexed for BM25, DPR, ANCE, BGE, Contriever, and ColBERT. For custom corpora, the CLI tool `rankify-index` supports on-demand index construction with FAISS and API-based embeddings.

3.2 Retriever Models

RANKIFY integrates **15 retrievers** behind a uniform interface. These span three families: (i) *Sparse*: BM25 (Robertson et al., 1995); (ii) *Dense*: DPR (Yates et al., 2021), ANCE (Xiong et al., 2021), ColBERT (Santhanam et al., 2021), BGE (Chen et al., 2023), Contriever (Izacard et al., 2021a), and others; (iii) *Reasoning-augmented*: Diver (Long et al., 2025), E5 (Wang et al., 2022b), InstructorL (Su et al., 2023), GritLM (Muennighoff et al., 2024), SFR (Meng et al., 2024), and related encoders that incorporate instruction-tuned or chain-of-thought reasoning into the retrieval step. When no local corpus is available, an online retrieval mode provides web-based evidence gathering via Craw4AI. Users initialize any retriever by specifying the model name, corpus (`index_type`: `wiki` or `msmarco`), and the number of documents to retrieve.

3.3 Re-Ranking Models

RANKIFY supports **24 re-ranking models** with **41 method variants** spanning three paradigms: (1) *Pointwise*: MonoBERT (Nogueira and Cho, 2019), MonoT5 (Nogueira et al., 2020), RankT5 (Zhuang et al., 2023), FlashRank, UPR, Inranker, Sentence-Transformer rankers, ColBERT-ranker, SPLADE, Transformer-Ranker, and TWOLAR. (2) *Pairwise*: PRP and EchoRank variants. (3) *Listwise*: RankGPT (Sun et al., 2023), ListT5, LiT5-Distill, RankVicuna, RankZephyr, FIRST, Blender, LLM-Layerwise, and In-Context Reranker.

All re-rankers share a single `Reranker` interface. Users specify the method, model name, and optional parameters (e.g., `top_k`, API key) and call `reranker.rerank(docs)`. This design supports both locally-hosted Hugging Face models and API-based services (e.g., OpenAI, Cohere) through the same code path. Additional code examples for common workflows are provided in Section D.

3.4 RAG Generation

RANKIFY's `Generator` module supports six RAG strategies: **Zero-shot**, **Basic RAG** (Lewis et al., 2020), **Chain-of-Thought** (Wei et al., 2022), **Fusion-in-Decoder (FiD)** (Izacard and Grave, 2020), **In-Context RALM** (Ram et al., 2023),

Retriever	Source	NQ (T20/T100)	WebQ (T20/T100)
DPR	Pyserini (Lin et al., 2021)	79.5 / 86.8	75.1 / 82.9
DPR	RANKIFY	79.5 / 86.8	75.1 / 82.9
BM25	Pyserini (Lin et al., 2021)	64.8 / 79.7	63.3 / 76.4
BM25	RANKIFY	64.8 / 79.7	63.3 / 76.4
Contriever	Official (Izacard et al., 2021b)	79.6 / 88.0	75.9 / 83.7
Contriever	RANKIFY	79.6 / 88.0	75.9 / 83.7
ColBERT	Official (Santhanam et al., 2021)	82.1 / 88.5	76.6 / 84.5
ColBERT	RANKIFY	82.1 / 88.5	76.6 / 84.5
ANCE	Official (Xiong et al., 2021)	82.1 / 87.9	–
ANCE	RANKIFY	82.5 / 88.5	76.6 / 84.3

Table 2: Retriever validation (Top-20 / Top-100 accuracy). RANKIFY matches Pyserini exactly for DPR and BM25, and closely matches official implementations for dense retrievers.

and **Selective Context**. Each strategy is implemented as a subclass of `BaseRAGMethod` with a common `answer_questions()` interface. Four inference backends are supported—Hugging Face Transformers, vLLM, LiteLLM, and OpenAI—accommodating both encoder-decoder (FiD) and decoder-only architectures. Users can switch between models (e.g., LLaMA (Touvron et al., 2023), GPT-4 (Achiam et al., 2023), T5 (Raffel et al., 2020)) and backends without changing pipeline code.

3.5 Evaluation

RANKIFY provides unified evaluation across all pipeline stages: (1) *Retrieval & re-ranking*: Top-k accuracy, nDCG, MAP, and MRR. (2) *Generation*: Exact Match, F1, precision, recall, and containment. (3) *RAG quality*: RAGAS-inspired metrics (Es et al., 2024)—**Faithfulness**, **Answer Relevancy**, and **Context Precision**—implemented via LLM-as-judge with configurable backends.

A single `Metrics` class computes all scores before and after re-ranking, enabling direct measurement of each component's contribution.

3.6 Agentic Recommendation and Deployment

Selecting an effective pipeline configuration requires balancing latency, GPU availability, and domain requirements. **RankifyAgent** is a conversational assistant that accepts natural-language requirements and returns (i) a justified recommendation and (ii) executable code that directly instantiates the suggested pipeline. Figure 2 shows a realistic interaction: given the constraint *"Need fast QA on CPU; prefer local models"*, the agent responds:

The agent connects to a configurable LLM backend (OpenAI, Azure, or a local model) and reasons over RANKIFY's module registry, which encodes each component's computational cost and quality profile. For production deployment, **RankifyServer** exposes any configured pipeline as a REST API. An interactive web playground (built on Streamlit) enables non-programmatic exploration: users can compare pipeline configurations side by side, inspect retrieved evidence, and export reproducible pipeline reports.

Re-ranker	Type	NQ-Test Top-1	NQ-Test Top-10	WebQ-Test Top-1	WebQ-Test Top-10
BM25 *(baseline)*	—	23.46	56.32	19.54	53.44
FlashRank (MiniLM)	Point.	34.70 ↑11.2	64.81 ↑8.5	31.84 ↑12.3	62.54 ↑9.1
RankT5-3B	Point.	47.17 ↑23.7	70.85 ↑14.5	40.40 ↑20.9	66.58 ↑13.1
TWOLAR-xl	Point.	46.84 ↑23.4	70.22 ↑13.9	41.68 ↑22.1	67.07 ↑13.6
EchoRank (Flan-T5-xl)	Pair.	41.68 ↑18.2	59.05 ↑2.7	36.22 ↑16.7	57.18 ↑3.7
LiT5-Distill-xl	List.	**47.81** ↑24.4	68.55 ↑12.2	**42.37** ↑22.8	65.55 ↑12.1
RankGPT (LLaMA-3.1-8B)	List.	41.55 ↑18.1	66.17 ↑9.9	38.77 ↑19.2	62.69 ↑9.3

Table 3: Re-ranking impact on BM25-retrieved documents (top-100) for NQ-Test and WebQ-Test (Top-k accuracy). One representative model per paradigm; ↑ = absolute gain over BM25 baseline.

4 Experiment

We evaluate RANKIFY along five axes: (i) implementation correctness, (ii) re-ranking on QA benchmarks, (iii) re-ranking on IR benchmarks, (iv) end-to-end pipeline quality, and (v) RAG strategies and online retrieval. Unless otherwise noted, experiments use a preprocessed English Wikipedia corpus (21 M passages, Dec. 2018 (Karpukhin et al., 2020)) and run on 2× NVIDIA A100 GPUs.

Implementation correctness. Before comparing models, we verify that RANKIFY reproduces established retrieval backends. Table 2 shows that RANKIFY exactly matches Pyserini for DPR and BM25 (same indexing backend), and closely matches official implementations for Contriever, ColBERT, and ANCE. This validation is important because all downstream results depend on the correctness of first-stage retrieval; the near-identical scores confirm that differences in later tables come from the re-rankers and generators, not from implementation drift. Extended validation results including TriviaQA are provided in Appendix A.

Re-ranking on QA benchmarks. Table 3 reports Top-k accuracy on NQ-Test and WebQ-Test when re-ranking BM25 top-100 candidates. We show one representative per paradigm (pointwise, pairwise, and listwise) to keep the demo paper

Re-ranker	Paradigm	DL19	DL20	COVID	SciFact	DBPedia	Robust04	Avg.
BM25 (first stage)	—	50.58	47.96	59.47	67.89	31.80	40.70	49.73
FlashRank (MiniLM)	Pointwise	70.40 ↑19.8	65.60 ↑17.6	69.66 ↑10.2	59.14 ↓8.8	39.75 ↑8.0	46.09 ↑5.4	58.44 ↑8.7
MonoT5-3B	Pointwise	71.83 ↑21.3	68.89 ↑20.9	80.71 ↑21.2	76.57 ↑8.7	44.45 ↑12.7	56.71 ↑16.0	66.53 ↑16.8
TWOLAR-xl	Pointwise	73.51 ↑22.9	**70.84** ↑22.9	82.70 ↑23.2	76.50 ↑8.6	**48.00** ↑16.2	57.90 ↑17.2	68.24 ↑18.5
Inranker-3B	Pointwise	72.71 ↑22.1	67.09 ↑19.1	81.75 ↑22.3	**78.31** ↑10.4	47.62 ↑15.8	**62.47** ↑21.8	68.33 ↑18.6
PRP (FLAN-UL2)	Pairwise	72.65 ↑22.1	70.46 ↑22.5	79.45 ↑20.0	73.33 ↑5.4	46.47 ↑14.7	53.43 ↑12.7	65.97 ↑16.2
Zephyr-7B	Listwise	74.22 ↑23.6	70.20 ↑22.2	80.70 ↑21.2	75.10 ↑7.2	43.10 ↑11.3	54.20 ↑13.5	66.25 ↑16.5
RankGPT (GPT-4)	Listwise	**75.50** ↑24.9	70.50 ↑22.5	**85.50** ↑26.0	74.90 ↑7.0	47.10 ↑15.3	57.50 ↑16.8	**68.50** ↑18.8

Table 4: Re-ranking effectiveness (nDCG@10) on TREC Deep Learning and BEIR benchmarks. First-stage retrieval: BM25 top-100. ↑ = gain; ↓ = regression vs. BM25 baseline. **Bold** = best per dataset. One representative model shown per paradigm from RANKIFY's 24 re-ranking models (41 method variants).

compact. Two patterns are consistent across both datasets. First, every re-ranker improves over the BM25 baseline, confirming that RANKIFY's re-ranking interfaces are effective regardless of model family. Second, stronger re-rankers primarily improve *early precision*: for example, on NQ-Test, Top-1 rises from 23.46 (BM25) to 47.81 with LiT5-Distill-xl, while on WebQ-Test it rises from 19.54 to 42.37. Lightweight models remain useful in practice: FlashRank (MiniLM) improves NQ Top-1 by ↑11.2 and WebQ Top-1 by ↑12.3, providing a CPU-friendly option with substantial gains. We also observe a trade-off between cost and quality: listwise and large pointwise models typically yield the strongest Top-1 results, while smaller models still improve Top-10 with lower latency. Full results across all 24 re-ranking models are provided in Appendix B.

Re-ranking on IR benchmarks. Table 4 extends the analysis to TREC Deep Learning and BEIR datasets using nDCG@10. The goal here is not only to show average gains, but also to demonstrate that RANKIFY supports heterogeneous re-rankers across multiple domains (biomedical, scientific, encyclopedic, and news). Re-ranking improves over BM25 on nearly all datasets and models, with especially large gains on DL19/DL20 and TREC-COVID. RankGPT (GPT-4) achieves the best scores on DL19 (75.50) and COVID (85.50), while TWOLAR-xl leads on DL20 (70.84) and DBPedia (48.00). Inranker-3B is competitive and even leads on SciFact (78.31) and Robust04 (62.47), indicating that the best re-ranker is dataset-dependent. We also retain one failure case (FlashRank on SciFact, ↓8.8) to show that RANKIFY surfaces negative results consistently rather than only reporting wins. The regression is informative: SciFact's claims contain specialized biomedical vocabulary that lightweight cross-encoders trained predominantly on web text struggle to score reliably, while larger pointwise re-rankers (MonoT5-3B, TWOLAR-xl, Inranker-3B) and listwise LLM re-rankers recover the performance and exceed BM25 by 8–13 nDCG points. A similar but milder pattern appears on Robust04, where smaller models lag larger counterparts by 5–16 nDCG points. Practitioners working in specialized domains should therefore prefer larger or domain-finetuned re-rankers, and RANKIFY makes this swap a one-line change.

Retriever	Reranker	EM	Faith.	Ans. Rel.	Ctx. Prec.
BM25	*None*	12.8	28.7	21.8	14.6
BM25	UPR	24.3 ↑11.5	34.8 ↑6.1	30.9 ↑9.1	20.3 ↑5.7
BM25	RankGPT	26.7 ↑13.9	35.8 ↑7.1	31.8 ↑10.0	23.1 ↑8.5
DPR	*None*	23.2	32.8	24.4	20.1
DPR	UPR	24.8 ↑1.6	35.9 ↑3.1	32.2 ↑7.8	25.5 ↑5.4
DPR	RankGPT	**29.9** ↑6.7	**37.8** ↑5.0	**35.3** ↑10.9	**27.8** ↑7.7

Table 5: End-to-end pipeline quality on NQ-Test (retrieve → rerank → generate with LLaMA-3.1-8B). ↑ = absolute gains over retrieval-only (*None*). Faithfulness, Ans. Rel., and Ctx. Prec. are RAGAS-inspired metrics (0–100 (Es et al., 2024)).

End-to-end pipeline quality. A core design goal of RANKIFY is *composability*: improvements in retrieval or re-ranking should transfer across generators without changing pipeline code. Figure 3 tests this directly by comparing three configurations (BM25 only, BM25+UPR, BM25+RankGPT) across six generator LLMs on NQ-Test. Re-ranking improves Exact Match for every generator, with RankGPT gains ranging from +8.1 pp (LLaMA-2-13B) to +16.4 pp (Gemma-2-2B). Notably, the weakest baseline generator (Mistral-7B at 11.2 EM) reaches 25.4 EM with RankGPT, a ∼2.3× improvement. This result is important for a demo setting because it shows users can benefit from the frame-

```python
from rankify.agent import RankifyAgent
agent = RankifyAgent(backend="openai")
resp = agent.chat(
    "Need fast QA on CPU; "
    "prefer local models."
)
print(resp.message)      # recommendation
print(resp.code_snippet) # runnable code
```

Output:

```
[Recommendation] Use BM25 + FlashRank (MiniLM-L12);
no GPU required, +11.2 Top-1 on NQ over BM25 alone.
Generator: LLaMA-3.1-8B (4-bit) via HuggingFace.

from rankify import pipeline
pipe = pipeline("rag",
    retriever="bm25", reranker="flashrank",
    generator="basic",
    model_name="meta-llama/Llama-3.1-8B",
    index_type="wiki", n_docs=10)
```

Figure 2: Querying RankifyAgent for a configuration recommendation: input query (top) and the agent's justified recommendation with ready-to-run code (bottom).

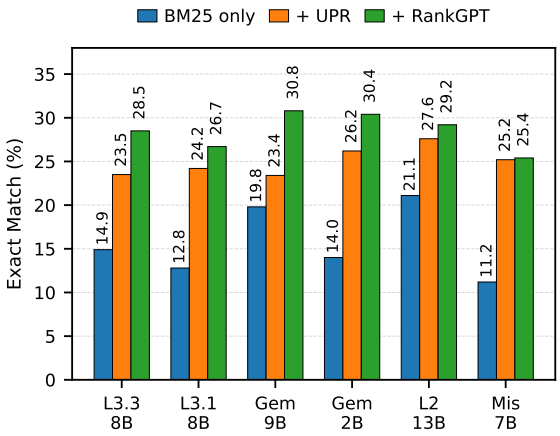

Figure 3: Exact Match on NQ-Test across six generator LLMs under three pipeline configurations (BM25 only, BM25+UPR, BM25+RankGPT). Re-ranking improves all generators, demonstrating RANKIFY's model-agnostic pipeline composability. L3.3/L3.1 = LLaMA 3.3/3.1 8B; Gem = Gemma-2; L2 = LLaMA-2 13B; Mis = Mistral-7B.

Dataset	Strategy	EM	F1	Prec	Rec	Con
NQ-Test	Basic RAG	**17.01**	**27.75**	**25.87**	**48.57**	**41.22**
	CoT	9.92	20.22	17.36	47.40	40.55
	Zero-shot	4.88	13.54	10.39	45.30	36.87
WebQ-Test	Basic RAG	**12.84**	**27.04**	**25.36**	52.23	**42.03**
	CoT	5.76	18.83	15.48	53.06	41.24
	Zero-shot	3.35	14.77	10.95	**54.93**	40.65

Table 6: Comparison of generation strategies with the same retriever and generator (DPR + LLaMA-3.1-8B).

work even with small local generators.

Table 5 complements Exact Match with RAGAS-inspired metrics (Es et al., 2024). The table shows that re-ranking improves not only answer correctness but also grounding quality. For BM25, RankGPT improves EM from 12.8 to 26.7, faithfulness from 28.7 to 35.8, and context precision from 14.6 to 23.1 (↑8.5). For DPR, the best configuration (DPR+RankGPT) reaches 29.9 EM and 37.8 faithfulness, with the highest answer relevancy and context precision as well. The larger gains for BM25 than for DPR are expected: weaker first-stage retrieval benefits more from second-stage re-ranking.

RAG strategies. Table 6 compares three generation strategies using DPR retrieval and LLaMA-3.1-8B. Basic RAG consistently yields the best EM/F1/precision, outperforming both Chain-of-Thought (CoT) and zero-shot generation (e.g., NQ EM: 17.0 vs. 9.9 vs. 4.9). Two effects explain this gap. First, CoT encourages the generator to produce intermediate reasoning steps that often drift from the retrieved evidence, lowering precision while keeping recall high (NQ recall: 47.40 for CoT vs. 48.57 for Basic RAG). Second, zero-shot generation discards the retrieved context entirely and relies on parametric knowledge, yielding the lowest precision across both datasets. This pattern indicates that for short-form factoid QA, grounding the generator directly on top-k contexts is more effective than additional reasoning prompts. RANKIFY's modular generator interface lets users switch among all six strategies (including FiD, In-Context RALM, and Selective Context) with a single argument, enabling broader strategy comparisons on tasks where reasoning-heavy prompting is more useful, e.g., multi-hop QA.

Online retrieval via Craw4AI. Table 7 validates RANKIFY's real-time web retrieval on 100 randomly sampled WebQ queries. Craw4AI achieves 90.0% Top-100 accuracy—matching BGE, the best offline dense retriever—without any pre-built index. While early precision is lower (34.0% Top-1 vs. 50.0% for BGE) due to diverse web results, Craw4AI narrows the gap at higher recall levels

Method	T@1	T@5	T@10	T@20	T@50	T@100
BM25 (offline)	20.0	36.0	46.0	62.0	72.0	80.0
DPR (offline)	42.0	**70.0**	**78.0**	78.0	82.0	86.0
BGE (offline)	**50.0**	66.0	76.0	**80.0**	**86.0**	**90.0**
Craw4AI (online)	34.0	56.0	60.0	76.0	**86.0**	**90.0**

Table 7: Online retrieval with Craw4AI is competitive with offline dense retrievers. It matches BGE at Top-50 and Top-100 on this 100-query WebQ subset, while trailing at Top-1/Top-10 where indexed dense retrieval remains stronger.

	NQ-Test		WebQ-Test	
Retriever	**R**	**+RankGPT**	**R**	**+RankGPT**
BM25	14.90	28.45	10.23	19.73
Contriever	15.29	30.55	10.67	19.78
DPR	28.08	**31.74**	19.83	**20.42**

Table 8: Retriever–reranker interaction (EM %) on NQ-Test and WebQ-Test using LLaMA V3.3 8B as the generator.

(86.0% at Top-50 vs. 86.0% for BGE). This validates RANKIFY's viability for real-time RAG applications where pre-built indices are unavailable.

Retriever–reranker interaction. Table 8 summarizes the interaction between first-stage retrievers and RankGPT re-ranking across three retrievers using LLaMA V3.3 8B on NQ-Test and WebQ-Test. DPR + RankGPT yields the best NQ configuration (31.7% EM) among the three retrievers shown. BM25 benefits most from re-ranking (↑13.6 pp on NQ), whereas DPR shows smaller but consistent gains (↑3.7 pp), confirming that weaker first-stage retrievers receive greater benefit from second-stage re-ranking. Across all 200+ tested configurations, re-ranking improves downstream EM by 5–15 % regardless of retriever or generator.

5 Conclusion

We presented RANKIFY, an open-source Python toolkit that unifies retrieval, re-ranking, and RAG generation in a single modular framework. RANKIFY provides 42 benchmark datasets with pre-retrieved documents and pre-built indices, 15 retrievers, 24 re-ranking models (41 variants spanning pointwise, pairwise, and listwise paradigms), and multiple RAG strategies across four inference backends. The experiments spanning 200+ configurations, which are facilitated by our framework, highlight the practical benefits of re-ranking, showing consistent gains in Exact Match (5–15%) and improvements in RAGAS context precision of up to 8.5 points across diverse pipeline setups.

Limitations. RANKIFY currently provides pre-built indices only for Wikipedia and MS MARCO, which may limit out-of-the-box applicability to specialized domains such as biomedical or legal corpora; users must run rankify-index to construct custom indices, adding overhead. RANKIFY's RAGAS-inspired evaluation metrics employ an LLM-as-judge paradigm, which itself introduces noise and potential bias depending on the backend model chosen; scores should therefore be interpreted as approximate indicators of generation quality rather than ground-truth measurements.

Acknowledgments

The authors would like to acknowledge the financial support provided by the Austrian Research Agency (FFG) for the project "AI Enabled Sustainability Jurisdiction Demonstrator" (project No. 915229).

The computational results presented have been achieved (in part) using the MUSICA HPC infrastructure.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Harrison Chase. 2022. Langchain: Building applications with llms through composability. Available at `https://www.langchain.com`.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2023. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2309.07597.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Benjamin Clavié. 2024. rerankers: A lightweight python library to unify ranking methods. *Preprint*, arXiv:2408.17344.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of Wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations*.

Hady ElSahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon S. Hare, Frédérique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Hiroaki Hayashi, Prashant Budania, Peng Wang, Chris Ackerson, Raj Neervannan, and Graham Neubig. 2020. Wikiasp: A dataset for multi-domain aspect-based summarization. *Transactions of the Association for Computational Linguistics (TACL)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

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*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021a. Unsupervised dense information retrieval with contrastive learning.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021b. Unsupervised dense information retrieval with contrastive learning. *arXiv:2112.09118*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *arXiv preprint arXiv:2405.13576*.

Mandar Joshi, Eunsol Choi, Daniel 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 (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, 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)*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, and 1 others. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. *Preprint*, arXiv:2004.12832.

Dongkyu Kim, Byoungwook Kim, Donggeon Han, and Matouš Eibich. 2024. Autorag: Automated framework for optimization of retrieval augmented generation pipeline. *Preprint*, arXiv:2410.20878.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur 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. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language*

Learning (CoNLL 2017), pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

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*, volume 33, pages 9459–9474.

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)*, pages 2356–2362.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Meixiu Long, Duolin Sun, Dan Yang, Junjie Wang, Yecheng Luo, Yue Shen, Jian Wang, Hualei Zhou, Chunxiao Guo, Peng Wei, and 1 others. 2025. Diver: A multi-stage approach for reasoning-intensive information retrieval. *arXiv preprint arXiv:2508.07995*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint*.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. Sfrembedding-mistral: enhance text retrieval with transfer learning. *Salesforce AI Research Blog*, 3(6).

Niklas Muennighoff, SU Hongjin, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. In *The Thirteenth International Conference on Learning Representations*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2017. MS MARCO: A human-generated MAchine reading COmprehension dataset.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713*.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA.

Bhawna Piryani, Jamshid Mozafari, and Adam Jatowt. 2024. Chroniclingamericaqa: A large-scale question answering dataset based on historical american newspaper pages. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2038–2048.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking. *arXiv preprint arXiv:2110.07367*.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, and 1 others. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. ASQA: Factoid questions meet long-form answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8273–8288, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agent. *ArXiv*, abs/2304.09542.

Simone Tedeschi, Simone Conia, Francesco Cecconi, and Roberto Navigli. 2021. Named Entity Recognition for Entity Linking: What works and what's next. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2584–2596, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*.

Jiexin Wang, Adam Jatowt, and Masatoshi Yoshikawa. 2022a. Archivalqa: A large-scale benchmark dataset for open-domain question answering over historical news collections. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3025–3035.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022b. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

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 *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William 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*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained transformers for text ranking: Bert and beyond. In *Proceedings of the 14th ACM International Conference on web search and data mining*, pages 1154–1156.

Nengjun Zhu, Jian Cao, Xinjiang Lu, and Qi Gu. 2021. Leveraging pointwise prediction with learning to rank for top-n recommendation. *World Wide Web*, 24:375–396.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2308–2313.

A Retriever Validation

Ensuring the correctness of first-stage retrieval is essential, as all downstream re-ranking and generation results depend on it. Table 9 extends the main-paper validation (Table 2) by including TriviaQA alongside NQ and WebQ. We compare RANKIFY's retriever implementations against three reference sources: the original paper implementations, Pyserini-based reproductions, and our own integrations.

For BM25 and DPR, RANKIFY uses Pyserini's indexing backend and therefore produces *identical*

scores. For dense retrievers (Contriever, ColBERT, and ANCE), minor deviations ($\leq$0.5 pp) arise from differences in tokenization or batch processing, but all results closely align with the official codebases. These results confirm that any performance differences observed in the re-ranking and generation experiments (Section 4) can be attributed to the re-rankers and generators, not to retrieval implementation artifacts.

Retriever	NQ	TriviaQA	WebQ
DPR (Karpukhin et al., 2020)	78.4 / 85.4	79.4 / 85.0	73.2 / 81.4
DPR_Pyserini (Lin et al., 2021)	79.5 / 86.8	79.7 / 85.1	75.1 / 82.9
DPR_RANKIFY	79.5 / 86.8	79.7 / 85.1	75.1 / 82.9
BM25 (Sachan et al., 2022)	63.0 / 78.2	76.4 / 83.1	62.3 / 75.5
BM25_Pyserini (Lin et al., 2021)	64.8 / 79.7	77.3 / 83.8	63.3 / 76.4
BM25_RANKIFY	64.8 / 79.7	77.3 / 83.8	63.3 / 76.4
Contriever (Izacard et al., 2021b)	79.6 / 88.0	80.4 / 85.7	75.9 / 83.7
Contriever_RANKIFY	79.6 / 88.0	80.3 / 85.7	75.9 / 83.7
ColBERT (Santhanam et al., 2021)	82.1 / 88.5	82.2 / 86.4	76.6 / 84.5
ColBERT_RANKIFY	82.1 / 88.5	82.2 / 86.4	76.6 / 84.5
ANCE (Xiong et al., 2021)	82.1 / 87.9	80.3 / 85.2	– / –
ANCE_RANKIFY	82.5 / 88.5	80.0 / 85.2	76.6 / 84.3

Table 9: Top-20 / Top-100 retrieval accuracy on NQ, TriviaQA, and WebQ. RANKIFY matches Pyserini results for DPR and BM25, and aligns with official implementations for ColBERT, Contriever, and ANCE.

B Full Re-Ranking Results

This section provides comprehensive re-ranking results across all models and method variants supported in RANKIFY. While the main paper (Tables 3 and 4) presents one representative model per paradigm to keep the presentation compact, Table 10 below reports the complete set of results for all 24 re-ranking models.

B.1 Full QA Re-Ranking Results

Table 10 reports re-ranking performance on BM25-retrieved documents (top-100) for NQ-Test and WebQ-Test across all supported models. Results include Top-1, Top-10, and Top-50 accuracy.

Several trends emerge beyond those discussed in the main paper. Within the UPR family, scaling from T5-Small to T0-3B yields a consistent monotonic improvement (e.g., NQ Top-1 rises from 23.60 to 35.42), confirming that larger unsupervised re-rankers produce better relevance estimates. Among the LiT5-Distill variants, the v2 models match or slightly exceed their v1 counterparts, while all xl variants perform comparably regardless of version. Notably, Inranker models show identical Top-1 scores across Small, Base, and 3B sizes (15.90 on NQ, 14.46 on WebQ), suggesting that the bottleneck for this architecture lies outside model scale. The ColBERT Ranker achieves the highest raw Top-1 on NQ (69.02) but shows anomalous Top-10 behavior (66.78), likely due to its late-interaction scoring being most effective for top-ranked candidates.

		NQ-Test			WebQ-Test		
Method	**Model**	**Top-1**	**Top-10**	**Top-50**	**Top-1**	**Top-10**	**Top-50**
BM25	–	23.46	56.32	74.57	19.54	53.44	72.34
UPR	T5-Small	23.60	59.97	75.15	18.55	55.56	72.68
	T5-Base	26.81	63.32	76.12	20.66	58.56	72.68
	T5-Large	29.75	65.67	76.48	24.21	60.38	73.32
	T0-3B	35.42	67.56	76.75	32.48	64.17	73.67
	GPT2	25.95	60.47	75.87	20.47	56.49	72.78
	GPT2-Medium	26.75	63.04	75.95	22.39	59.54	72.44
	GPT2-Large	26.59	62.68	75.95	24.06	59.84	72.78
	GPT-Neo-2.7B	28.75	64.81	76.56	24.75	59.64	72.63
RankGPT	LLaMA-3.1-8B	41.55	66.17	75.42	38.77	62.69	73.12
FlashRank	TinyBERT	31.49	61.57	74.95	28.54	60.62	73.17
	MiniLM-L12	34.70	64.81	76.17	31.84	62.54	73.47
	T5-Flan	7.95	36.14	66.67	12.05	42.96	67.27
RankT5	T5-Base	43.04	68.47	76.28	36.95	64.27	74.45
	T5-Large	45.54	70.02	76.81	38.77	66.48	74.31
	T5-3B	47.17	70.85	76.89	40.40	66.58	74.45
Inranker	Small	15.90	46.84	69.83	14.46	46.25	69.98
	Base	15.90	48.11	69.66	14.46	46.80	69.68
	3B	15.90	48.06	69.00	14.46	46.11	69.34
LLM2Vec	LLaMA-3.1-8B	24.32	59.55	75.26	26.72	60.48	73.47
MonoBERT	Large	39.05	67.89	76.56	34.99	64.56	73.96
TWOLAR	XL	46.84	70.22	76.86	41.68	67.07	74.40
EchoRank	Flan-T5-Large	36.73	59.11	62.38	31.74	58.75	61.51
	Flan-T5-XL	41.68	59.05	62.38	36.22	57.18	61.51
InContext	LLaMA-3.1-8B	15.15	57.11	76.48	18.89	52.16	71.70
LiT5	Distill-base	40.05	65.95	75.73	36.76	63.48	73.12
	Distill-large	44.40	67.59	76.01	39.66	64.56	73.67
	Distill-xl	47.81	68.55	76.26	42.37	65.55	73.62
	Distill-base-v2	42.57	66.73	75.56	39.61	64.22	73.32
	Distill-large-v2	46.53	67.83	75.87	41.97	65.64	72.98
	Distill-xl-v2	47.92	69.03	76.17	41.53	65.69	73.27
Sentence Transformer	GTR-T5-Base	39.41	65.95	76.03	36.56	64.32	73.62
	GTR-T5-Large	40.63	68.25	76.73	38.97	65.30	73.57
	GTR-T5-XL	41.55	67.78	76.81	38.92	66.04	74.01
	GTR-T5-XXL	42.93	68.55	77.00	39.41	65.89	74.01
ColBERT Ranker v2		69.02	66.78	–	33.70	35.51	–

Table 10: Full re-ranking results on BM25-retrieved documents (top-100) for NQ-Test and WebQ-Test. Top-1, Top-10, and Top-50 accuracy reported. The main paper Table 3 shows one representative per paradigm; this table provides the complete set.

C Datasets Statistics

Table 11 provides detailed statistics for all 42 datasets supported in RANKIFY, organized by task category. The collection spans 12 categories—open-domain QA (17 datasets), multi-hop QA (4), temporal QA (2), long-form QA (2), multiple-choice (6), entity linking (3), slot filling (2), dialogue (1), fact verification (1), summarization (1), IR benchmarks (2), and specialized evaluation (1)—totaling over 6 million examples.

Each dataset is distributed with 1 000 pre-retrieved documents per query (obtained via BM25, DPR, and ColBERT) and pre-built FAISS indices, enabling researchers to begin re-ranking and generation experiments immediately without the time-consuming indexing step. Dataset sizes range from

Task	Dataset	# Train	# Val	# Test
QA	NQ	79,168	8,757	3,610
	TriviaQA	78,785	8,837	11,313
	WebQ	3,778	–	2,032
	SQuAD	87,599	10,570	–
	NarrativeQA	32,747	3,461	10,557
	MSMARCO-QA	808,731	101,093	–
	PopQA	–	–	14,267
	SIQA	33,410	1,954	–
	Fermi	8,000	1,000	1,000
	WikiQA	20,360	2,733	6,165
	AmbigQA	10,036	2,002	–
	CommonsenseQA	9,741	1,221	–
	PIQA	16,113	1,838	–
	BoolQ	9,427	3,270	–
	StrategyQA	2,290	–	–
	CuratedTREC	430	–	430
	WikiPassageQA	3,332	416	416
Multi-Hop QA	2WikiMultiHopQA	15,000	12,576	–
	Bamboogle	–	–	125
	Musique	19,938	2,417	–
	HotpotQA	90,447	7,405	–
Temporal QA	ArchivalQA	384,426	48,304	48,760
	ChroniclingQA	385,629	21,739	21,735
Long-Form QA	ELI5	272,634	1,507	–
	ASQA	4,353	948	–
Multiple-Choice	MMLU	99,842	1,531	14,042
	TruthfulQA	–	817	–
	HellaSwag	39,905	10,042	–
	ARC	3,370	869	3,548
	OpenBookQA	4,957	500	500
	QuaRTz	2,696	384	784
Entity Linking	WNED	–	8,995	–
	AIDA CoNLL-YAGO	18,395	4,784	–
	EntityQuestions	–	–	2,472
Slot Filling	Zero-shot RE	147,909	3,724	–
	T-REx	2,284,168	5,000	–
Dialog	WOW	63,734	3,054	–
Fact Verif.	FEVER	104,966	10,444	–
Summariz.	WikiAsp	300,636	37,046	37,368
IR	BEIR	–	–	6,980
	DL19/20	–	–	43/54
Specialized	DomainRAG	–	–	485

Table 11: Dataset statistics for all 42 datasets in RANKIFY. Total: 6M+ examples across 12 task categories. Each dataset ships with 1 000 pre-retrieved documents and pre-built indices.

125 examples (Bamboogle) to over 2.2 million (T-REx), providing evaluation at multiple scales. Where official train/val/test splits exist, we preserve them; datasets without a standard test split (marked "–") are typically used for training or validation only.

D Code Examples

This section provides additional code examples illustrating common RANKIFY workflows beyond the `pipeline()` API shown in Figure 1. Each snippet is self-contained and demonstrates a distinct use case: loading pre-retrieved datasets (Listing 1), applying a specific re-ranking model (Listing 2), constructing a full retrieve–rerank–generate pipeline with evaluation (Listing 3), and deploying a configured pipeline as a REST service (Listing 4). Together, these examples show that RANKIFY enables end-to-end experimentation and deployment with minimal boilerplate code.

```python
from rankify.dataset.dataset import Dataset
dataset = Dataset(
    retriever="bm25",
    dataset_name="nq-test"
)
documents = dataset.download(
    force_download=False
)
print(f"Loaded {len(documents)} queries")
print(f"First query: {documents[0].question}")
```

Listing 1: Loading a pre-retrieved dataset.

```python
from rankify.rerankers.reranker import Reranker
reranker = Reranker(
    method="monot5",
    model_name="monot5-3b",
    top_k=10
)
reranked = reranker.rerank(documents)
```

Listing 2: Re-ranking with MonoT5.

```python
from rankify import pipeline
from rankify.metrics import Metrics

# Build pipeline
pipe = pipeline(
    "rag",
    retriever="dpr",
    reranker="rankgpt",
    generator="basic",
    model_name="meta-llama/Llama-3.1-8B",
    index_type="wiki",
    n_docs=10,
)

# Run on dataset
results = pipe(documents)

# Evaluate
metrics = Metrics()
scores = metrics.evaluate(
    results,
    metrics=["em", "f1", "faithfulness"]
)
print(scores)
```

Listing 3: End-to-end pipeline with evaluation.

```python
from rankify.server import RankifyServer
server = RankifyServer(
    retriever="bge",
    reranker="flashrank",
    generator="basic",
    model_name="meta-llama/Llama-3.1-8B",
)
server.run(host="0.0.0.0", port=8080)
# POST /rerank, /generate, /pipeline
```

Listing 4: Exposing a pipeline via RankifyServer.