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

Anonymous ACL submission

Abstract

Retrieval, re-ranking, and retrieval-augmented generation (RAG) are critical for modern information retrieval and question answering systems. However, existing solutions are fragmented, lacking unified frameworks for consistent evaluation and experimentation. We introduce **RANKIFY**, a modular open-source toolkit that unifies retrieval, re-ranking, and RAG in a cohesive framework. RANKIFY supports diverse retrieval methods (sparse, dense, and online via Craw4AI), 24 re-ranking models with 41 methods spanning pointwise, pairwise, and listwise approaches, and advanced RAG techniques. It provides 42 pre-retrieved datasets with pre-built indices, RAGAS-inspired evaluation metrics for holistic assessment, and auto-indexing with FAISS and API embeddings. RANKIFY is available on PyPI[1], GitHub[2], with comprehensive documentation[3], enabling scalable and reproducible research.

1 Introduction

Information retrieval (IR) systems (Singhal et al., 2001; Chowdhury, 2010) are fundamental to many applications (Wang et al., 2024a; Liu et al., 2024), including question-answering (Khamnuansin et al., 2024; Abdallah et al., 2025), search engines (Croft et al., 2010; Xiong et al., 2024), and knowledge-based generation (Long et al., 2024). These systems often rely on a two-stage pipeline: a retriever that efficiently identifies a set of candidate documents and a re-ranker that refines these results to maximize relevance to the query (Yates et al., 2021; Nogueira et al., 2019; Ren et al., 2021). This approach has proven highly effective, with retrieval and re-ranking methods achieving state-of-the-art performance across diverse NLP benchmarks.

Retrievers are central to IR, encompassing sparse, dense, and hybrid methods. Sparse retrievers like BM25 (Robertson et al., 1995) use term matching but struggle with semantic relationships. Dense retrievers, such as DPR (Yates et al., 2021), ColBERT (Santhanam et al., 2021), and BGE (Chen et al., 2023), encode queries and documents into semantic vectors, while hybrid methods (Yang et al., 2019; Chen et al., 2022) combine both strengths. RANKIFY now supports real-time online retrieval via Craw4AI and auto-indexing with FAISS and API embeddings (e.g., GPT) for efficient, scalable retrieval.

Re-ranking enhances retrieval by prioritizing relevant documents. Pointwise methods (Nogueira et al., 2019; Abdallah et al., 2025) assign independent scores, pairwise methods (Huang and Chen, 2024; Qin et al., 2023) compare document pairs, and listwise methods (Sun et al., 2023; Yoon et al., 2024) optimize entire document lists using LLMs, often with sliding window strategies to handle input limitations.

Finally, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Gao et al., 2023) improves language models by integrating retrieved documents for knowledge-intensive tasks. Existing toolkits like Pyserini (Lin et al., 2021) and Rerankers (Clavié, 2024) lack flexibility and require extensive preprocessing. RANKIFY unifies retrieval, re-ranking, and advanced RAG methods (e.g., FiD, in-context learning) in a modular framework, offering pre-retrieved datasets and comprehensive evaluation tools for streamlined research.

The main contributions of this paper are as follows: (1) **Curated Datasets and Indexing**: RANKIFY provides 42 datasets with 1,000 pre-retrieved documents per query and precomputed Wikipedia/MS MARCO indices, plus auto-indexing with FAISS and API embeddings. (2) **Diverse Retrieval and Re-ranking methods**: It

[1] https://pypi.org/project/rankify/
[2] https://github.com/
[3] http://readthedocs.io/

Feature	FlashRAG	AutoRAG	FastRAG	Rerankers	RANKIFY	Advantage
Dataset Support						
# Datasets	38	4	0	0	**42**	+4 over FlashRAG
Pre-retrieved docs	✗	✗	✗	✗	✓	Faster experimentation
Pre-built indices	✗	✗	✗	✗	✓	No indexing overhead
Online retrieval	✗	✗	✗	✗	✓	Real-time capabilities
Model Support						
# Retrievers	3	2	1	0	**7**	+4 over FlashRAG
# Re-rankers (methods)	2	7	3	15	**24 (41)**	+9 models, +26 methods
Pointwise re-ranking	Limited	✓	✓	✓	✓	–
Pairwise re-ranking	✗	Limited	✗	Limited	✓	Only comprehensive support
Listwise re-ranking	✗	✗	✗	Limited	✓	Only comprehensive support
# RAG methods	16	1	7	0	9	Focused on core methods
Evaluation Capabilities						
Retrieval metrics	✓	✓	✗	✓	✓	–
Re-ranking metrics	Limited	✓	✗	✓	✓	–
RAG metrics (EM, F1)	✓	✓	✗	✗	✓	–
RAGAS-inspired metrics	✗	✗	✗	✗	✓	Faithfulness, context metrics
Full pipeline evaluation	Limited	Limited	✗	✗	✓	End-to-end benchmarking
Design Philosophy						
Training support	✗	✗	✗	✗	✗	Future work
Benchmarking focus	✓	Limited	✗	✓	✓	–
Reproducibility	Limited	Limited	Limited	✓	✓	Pre-computed resources
Modularity	✓	✓	✓	✓	✓	–
Documentation	✓	✓	Limited	✗	✓	–

Table 1: Detailed comparison of RANKIFY with existing IR and RAG toolkits. The **Advantage** column quantifies RANKIFY's benefits (e.g., "+4 datasets") or highlights exclusive capabilities. 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.

supports sparse (BM25), dense (DPR, ColBERT, etc.), and online (Craw4AI) retrievers, with 24 re-ranking models for flexible experimentation. (3) **RAG and Evaluation**: The framework integrates retrieval, re-ranking, and advanced RAG methods, with metrics for consistent benchmarking. (4) **Accessibility**: RANKIFY is freely available on PyPI and GitHub with extensive documentation for easy adoption.

RANKIFY builds upon and extends existing retrieval and RAG toolkits. While frameworks like FlashRAG (Jin et al., 2024) (38 datasets, 3 retrievers) and AutoRAG (Kim et al., 2024) (4 datasets, 7 re-rankers) provide valuable functionality, they lack comprehensive support for pairwise/listwise re-ranking, pre-retrieved datasets with pre-built indices, and RAGAS-inspired evaluation metrics. Rerankers (Clavié, 2024) focuses solely on re-ranking without retrieval or RAG integration. RANKIFY uniquely combines these capabilities (see Table 1 for detailed comparison and Appendix A for extended discussion of related work).

2 RANKIFY

Existing retrieval and ranking solutions are fragmented, requiring researchers to integrate multiple toolkits for experimentation. RANKIFY addresses this by providing an end-to-end ecosystem unifying retrieval, re-ranking, and RAG within a modular, extensible Python framework. Built using PyTorch (Paszke et al., 2019), TensorFlow (Abadi et al., 2016), Spacy (Honnibal et al., 2020), and Pyserini (Lin et al., 2021), it is available on PyPI[4]:

```
$ pip install rankify
```

RANKIFY supports both pre-retrieved datasets and real-time retrieval via Craw4AI, with ready-to-use indices for Wikipedia and MS MARCO. The architecture comprises four modules (Figure 1 and 6): **(1) Datasets**: 42 standardized datasets (NQ, TriviaQA, HotpotQA, etc.) with custom dataset support. **(2) Retrievers**: BM25, DPR, ANCE, BGE, ColBERT, and online retrieval methods. **(3) Re-Rankers**: 24 models with 41 methods spanning pointwise, pairwise, and listwise approaches. **(4) RAG**: Multiple backends (HuggingFace, VLLM) supporting in-context learning, FiD, and RALM. The unified API abstracts implementation details, enabling seamless experimentation. RANKIFY is open-source[5] with comprehensive documentation[6].

[4] https://pypi.org/project/rankify/
[5] https://github.com/abdoelsayed2016/reranking
[6] http://rankify.readthedocs.io/

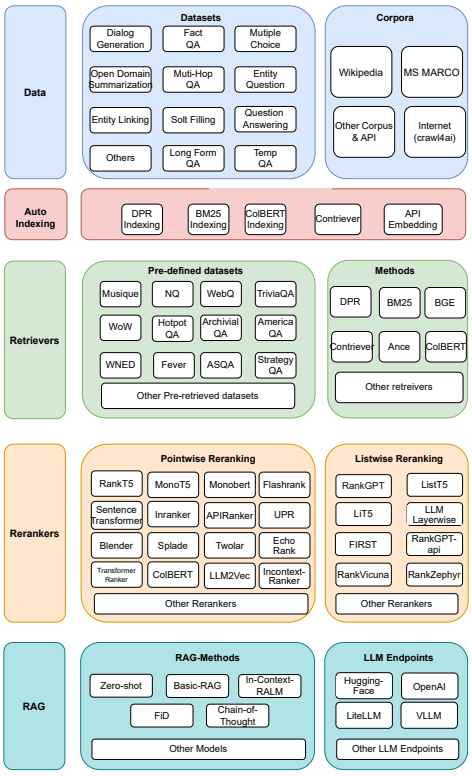

Figure 1: The architecture of the RANKIFY, showing the interplay between its core modules: *Datasets*, *Retrievers*, *Re-Rankers*, and *RAG Evaluation*. Each module operates independently while seamlessly integrating with others, enabling end-to-end retrieval and ranking workflows.

2.1 Prebuilt Retrieval Corpora and Indexes

RANKIFY provides two large-scale retrieval corpora: **Wikipedia** (Karpukhin et al., 2020) and **MS MARCO** (Nguyen et al., 2017), enabling researchers to conduct retrieval experiments on well-established benchmarks. To support efficient retrieval, it includes pre-built index files for each retriever, allowing users to directly query the corpora without requiring costly indexing. Each corpus has been indexed for multiple retrieval models, including BM25, DPR, ANCE, BGE, Contriever, and ColBERT. By providing precomputed indexes, RANKIFY simplifies large-scale retrieval research while maintaining consistency across different retrieval pipelines. Users can view all available datasets using a single command (see Listing 1 in Appendix G).

2.2 Datasets

RANKIFY provides a standardized schema (**Question**, **Answers**, **Contexts**) for 42 datasets across 9 task categories: QA (17 including NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), SQuAD (Rajpurkar et al., 2016), WebQuestions (Berant et al., 2013), NarrativeQA (Kočiskỳ et al., 2018), MSMARCO-QA (Nguyen et al., 2017), PopQA (Mallen et al., 2022), SIQA (Sap et al., 2019), Fermi (Kalyan et al., 2021), WikiQA (Yang et al., 2015), AmbigQA (Min et al., 2020), CommonsenseQA (Talmor et al., 2019), PIQA (Bisk et al., 2019), BoolQ (Clark et al., 2019), StrategyQA (Geva et al., 2021), CuratedTREC (Baudiš and Šedivỳ, 2015), WikiPassageQA (Cohen et al., 2018)), Multi-Hop QA (4 datasets) (Ho et al., 2020; Press et al., 2023; Trivedi et al., 2022; Yang et al., 2018), 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., 2021b,a; Lin et al., 2022; Zellers et al., 2019; Clark et al., 2018; Mihaylov et al., 2018; Tafjord et al., 2019), Entity-linking (Tedeschi et al., 2021; Hoffart et al., 2011; Sciavolino et al., 2021), Slot Filling (Levy et al., 2017; ElSahar et al., 2018), plus dialog (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)). For complete listings, the reader should see Appendix B.

Each dataset includes 1,000 pre-retrieved documents from Wikipedia/MS MARCO (processed by BM25, DPR, ColBERT) with prebuilt FAISS indices and API-based embeddings for rapid experimentation. Online retrieval via Craw4AI enables dynamic expansion beyond offline corpora. Users load datasets via `load_dataset`/`load_dataset_qa` functions or create custom datasets (Listings 1–4, Appendix G).

2.3 Retriever Models

RANKIFY supports diverse retrievers via a unified interface: sparse methods (BM25 (Robertson et al., 1995)) and dense models (DPR (Yates et al., 2021), ANCE (Xiong et al., 2021), BGE (Chen et al., 2023), Contriever (Izacard et al., 2021a), ColBERT (Khattab and Zaharia, 2020)). Users initialize retrievers by specifying the model, corpus (`index_type`: `msmarco` or `wiki`), and document count, as shown in Listing 5 (Appendix G). This flexibility enables experimentation across different retrieval strategies and task requirements.

Dataset	Split	Top-1	Top-5	Top-10	Top-20	Top-50	Top-100	Dataset	Split	Top-1	Top-5	Top-10	Top-20	Top-50	Top-100
2WikiMultiHopQA	Train	8.85	19.78	28.15	38.16	51.16	57.70	ArchivalQA	Val	18.18	33.29	39.72	45.96	53.68	58.74
	Val	17.17	31.72	40.14	47.73	55.88	61.17		Test	17.68	32.55	39.26	45.36	52.76	58.08
ChroniclingAmericaQA	Val	4.26	11.80	16.42	21.50	28.19	33.15	AmbigQA	Train	27.30	50.67	59.63	67.63	75.68	79.87
	Test	4.18	11.23	15.83	20.87	27.49	32.48		Val	30.22	54.25	65.58	72.98	80.47	84.92
ARC	Train	4.45	18.16	31.39	51.39	78.19	91.36	TriviaQA	Train	47.87	67.22	72.90	77.15	81.30	83.64
	Val	4.83	19.45	30.84	52.13	79.06	91.83		Val	48.70	67.57	72.84	77.38	81.19	83.59
	Test	4.96	19.84	33.57	51.94	79.09	91.80		Test	48.22	67.44	72.79	77.28	81.42	83.87
SQuAD	Train	31.84	50.24	57.15	63.33	70.32	74.61	MMLU	Train	3.65	14.14	23.98	38.68	62.22	78.04
	Val	36.94	57.61	64.78	71.31	77.89	81.96		Val	6.53	21.75	33.64	50.62	75.77	88.18
	Test	36.67	57.37	64.50	71.09	77.74	81.84		Test	6.53	22.01	34.67	51.22	74.35	86.40
QuaRTz	Train	7.81	26.82	40.62	63.54	84.38	92.19	NarrativeQA	Train	16.80	25.24	28.46	31.50	35.74	38.65
	Val	7.81	26.82	40.62	63.54	84.38	92.19		Val	15.46	23.92	27.10	30.45	35.08	38.89
	Test	5.87	23.60	38.90	58.67	80.74	90.43		Test	16.80	25.24	28.46	31.50	35.74	38.65
NQ	Train	22.89	44.97	54.32	62.73	71.75	77.03	OpenBookQA	Train	3.35	16.24	28.99	47.85	75.01	88.56
	Val	23.62	45.60	55.41	64.19	72.51	77.72		Val	3.40	15.20	31.60	50.60	79.00	90.00
	Test	23.46	45.62	56.32	64.88	74.57	79.72		Test	4.00	16.80	32.20	51.20	74.00	88.80
WebQuestions	Train	21.41	44.71	55.35	64.21	72.71	78.45	Musique	Train	6.23	13.70	18.54	24.39	33.77	40.81
	Test	19.54	42.86	53.44	63.25	72.34	76.43		Val	7.36	15.47	21.39	25.98	34.59	41.29
T-REx	Val	45.48	64.34	70.60	76.04	80.42	82.98	TruthfulQA	Val	2.33	7.59	12.12	20.69	38.19	56.55
SIQA	Train	0.34	1.52	2.78	4.85	8.92	13.43	HotpotQA	Train	34.89	51.71	58.13	64.09	71.11	75.49
	Val	0.61	2.05	3.58	6.55	11.31	16.58		Val	28.13	45.02	51.92	57.79	65.44	70.67
StrategyQA	Train	4.19	15.72	26.29	39.87	52.88	57.86	PopQA	Test	25.00	38.77	44.70	49.74	56.82	62.36
WOW	Train	0.26	0.38	0.41	0.45	0.49	0.52	ZSRC	Train	50.53	67.23	72.06	76.10	80.35	82.99
	Val	0.20	0.33	0.36	0.39	0.46	0.46		Val	52.26	70.86	76.29	81.44	85.39	87.73
Bamboogle	Test	5.60	12.80	17.60	24.80	39.20	44.00	EntityQuestions	Test	43.36	60.50	66.08	70.61	75.41	79.06

Table 2: Retrieval performance of BM25 across several selected datasets. The table reports Top-1, Top-5, Top-10, Top-20, Top-50, and Top-100 retrieval accuracy for each dataset split.

2.4 Re-Ranking Models

In information retrieval, two-stage pipelines retrieve candidate documents with an efficient retriever, then they re-rank them using neural models to capture deeper query-document semantics. Re-ranking models vary in architecture and trade-offs, complicating method selection. RANKIFY offers a unified interface for re-ranking, supporting 24 models with 41 sub-methods, including pointwise (MonoBERT (Nogueira and Cho, 2019), MonoT5 (Nogueira et al., 2020), RankT5 (Zhuang et al., 2023)), pairwise, and listwise (RankGPT (Sun et al., 2023)) approaches. Auto-indexing with FAISS and API embeddings (e.g., GPT) optimizes performance, while Craw4AI enables dynamic document retrieval. Users can apply models like ColBERT (Santhanam et al., 2021) or API-based RankGPT via a simple API (Listing 6, Appendix G). This flexibility, supporting Hugging Face and API-based models, ensures adaptable re-ranking for diverse tasks.

2.5 RAG Models

Retrieval-Augmented Generation (RAG) enhances language models by generating responses based on dynamically retrieved documents, effective for knowledge-intensive tasks like open-domain QA and fact verification. RANKIFY's Generator module (Figure 5, Appendix E) supports multiple RAG methods: zero-shot generation, basic RAG (Lewis et al., 2020), Chain-of-Thought (Wei et al., 2022), Fusion-in-Decoder (FiD) (Izacard and Grave, 2020), and in-context learning (Ram et al., 2023). The unified interface supports four inference backends (HuggingFace, OpenAI, vLLM, LiteLLM) and diverse LLMs (LLaMA (Touvron et al., 2023), GPT-4 (Achiam et al., 2023), T5 (Raffel et al., 2020)), accommodating both encoder-decoder (FiD) and decoder-only architectures (Listing 7, Appendix G).

2.6 Evaluation Metrics

RANKIFY provides comprehensive evaluation metrics across all pipeline stages. For retrieval and re-ranking, it supports Top-k accuracy, nDCG, MAP, and MRR (see Listing 8 in Appendix G). For generation quality assessment, RANKIFY includes both traditional QA metrics—Exact Match (EM), F1, precision, recall, and containment—and modern RAG-specific evaluation frameworks. To address the growing need for holistic RAG evaluation beyond simple answer matching, RANKIFY integrates RAGAS-inspired metrics (Es et al., 2024)[7]. These metrics assess three critical dimensions: **Faithfulness**, **Answer Relevancy**, and **Context Precision**. RANKIFY implements these metrics using LLM-as-judge approaches with configurable backends (OpenAI GPT-4, LLaMA-3.1, etc.) and embedding models for semantic similarity computation.

3 Experimental Results and Discussion

As stated before, RANKIFY enables benchmarking of retrieval, re-ranking, and RAG methods, supporting custom evaluations and optimizations. Based

[7]RAGAS: Retrieval-Augmented Generation Assessment, https://github.com/explodinggradients/ragas

Retriever	Dataset	Top-1	Top-5	Top-10	Top-20	Top-50	Top-100
Contriever (Izacard et al., 2021a)	NQ	38.81	65.65	73.91	79.56	84.88	88.01
	WebQ	35.97	63.83	69.78	75.94	81.64	83.76
	TriviaQA	49.85	71.39	76.68	80.26	83.85	85.72
DPR (Yates et al., 2021)	NQ	44.57	67.76	74.52	79.50	84.40	86.81
	WebQ	44.64	63.98	70.52	75.05	80.51	82.97
	TriviaQA	57.47	72.40	76.50	79.76	82.96	85.09
ColBert (Santhanam et al., 2021)	NQ	42.99	68.78	76.12	82.08	86.15	88.59
	WebQ	40.11	64.81	71.56	76.57	81.20	84.50
	TriviaQA	57.36	75.90	79.49	82.18	84.74	86.41
Ance (Xiong et al., 2021)	NQ	50.80	71.86	78.12	82.52	86.23	88.28
	WebQ	46.51	66.24	71.80	76.62	81.79	84.25
	TriviaQA	56.46	72.14	76.65	80.04	83.37	85.22
BGE (Xiao et al., 2023)	NQ	48.03	72.22	78.50	82.66	86.90	89.45
	WebQ	42.67	65.45	72.54	79.04	82.78	85.88
	TriviaQA	57.81	75.39	79.83	82.92	85.53	86.94

Table 3: Comparison of different retrieval models on NQ, WebQ, and TriviaQA. The table reports Top-1 to Top-100 accuracy for each retriever.

Retrievers	NQ	TriviaQA	WebQ
DPR (Yates et al., 2021)	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 / 84.25

Table 4: Top-20/Top-100 retrieval accuracy on NQ, TriviaQA, and WebQ. RANKIFY matches Pyserini results as it uses Pyserini's indexing backend.

on it, we conduct extensive experiments to demonstrate its usefulness, to provide a novel performance reference and to compare against the results of original implementations.

3.1 Experiment Setup

The experiments were done on 2x NVIDIA A100 GPUs. We used a preprocessed English Wikipedia dump (December 2018, (Yates et al., 2021)), split into 100-word passages (over 21 million total), as the primary corpus. Auto-indexing with FAISS and API embeddings (e.g., GPT) was used to streamline processing. Retrieval was evaluated on datasets like NQ, TriviaQA, HotpotQA, and ArchivalQA, with Top-k accuracy (k=1, 5, 10, 20, 50, 100). For re-ranking we used MSMARCO, WebQuestions, and PopQA, while Craw4AI was used for real-time retrieval. We tested BM25, DPR, ANCE, BGE, and ColBERT for retrieval; MonoT5, RankT5, and RankGPT for re-ranking; and zero-shot, FiD, and RALM for RAG, showcasing RANKIFY 's adaptability.

3.2 Retrieval Results

BM25 retrieval performance across 24 dataset splits is shown in Table 2. Performance varies significantly by task: open-domain QA datasets (TriviaQA, Natural Questions) achieve high Top-k accuracy, while multi-hop datasets (2WikiMultiHopQA, HotpotQA) show lower retrieval rates, reflecting the challenge of retrieving evidence across multiple documents. Temporal datasets (ArchivalQA, ChroniclingAmericaQA) exhibit moderate performance due to their time-sensitive content.

Table 3 presents dense retriever performance on NQ, WebQ, and TriviaQA. MSS-DPR achieves the highest Top-1 accuracy: 50.16% (NQ), 44.24% (WebQ), and 61.63% (TriviaQA), demonstrating fine-tuned dense retrieval effectiveness. Standard DPR performs strongly on NQ (48.67%) and TriviaQA (57.47%). ColBERT shows competitive performance at higher recall (Top-50, Top-100), while dense methods (DPR, MSS-DPR, ColBERT) consistently outperform MSS and Contriever across all metrics.

3.3 Comparison with Original Implementations

To validate RANKIFY's retrieval module, we compare its performance against original implementations and Pyserini-based baselines for DPR, Contriever, BM25, and ColBERT on NQ, TriviaQA, and WebQ datasets, using Top-20 and Top-100 accuracy. Table 4 shows RANKIFY matches Pyserini's performance for DPR and BM25, leveraging Pyserini's optimized indexing. For Contriever and ColBERT, our independent implementations align closely with original results, confirming correctness. RANKIFY 's BM25 and DPR outperform original baselines due to Pyserini's indexing optimizations. ColBERT and Contriever results replicate official implementations.

3.4 Re-Ranking Results

Table 7 presents the performance of re-ranking methods applied to BM25-retrieved documents for NQ-Test and WebQ datasets, reporting Top-1, Top-10, and Top-50 accuracy. Re-ranking consistently enhances retrieval quality, with significant Top-1 accuracy gains over the BM25 baseline. Transformer-based models like RankT5-3B achieve the highest performance, with 47.17% Top-1 accuracy on NQ-Test and 40.40% on WebQ, while RankGPT (LLaMA-3.1-8B) improves WebQ Top-1 from 18.80% (BM25) to 38.77%. Lighter models,

Method	Model	DL19	DL20	Covid	NFCorpus	Touche	DBPedia	SciFact	Signal	News	Robust04
UPR	T5-Small	49.94	47.07	62.81	30.46	21.54	31.98	61.68	29.71	27.41	33.14
	T5-Base	55.12	53.94	64.12	31.12	19.42	34.28	65.56	30.50	27.32	33.40
	T5-Large	58.33	56.05	68.69	31.71	18.94	35.35	65.44	32.80	25.25	34.48
	T0-3B	60.18	59.55	68.83	33.48	23.97	34.41	71.21	33.02	39.10	41.74
FlashRank	TinyBERT	67.68	60.65	61.48	32.85	32.72	36.04	64.08	31.85	37.53	41.37
	MiniLM	70.40	65.60	69.66	32.80	34.61	39.75	59.14	28.10	41.44	46.09
MonoT5	Base-10k	71.38	66.31	74.61	35.69	37.86	42.09	73.39	32.14	46.09	51.69
	Large-10k	72.12	67.11	77.38	36.91	38.31	41.55	73.67	33.17	47.54	56.12
	3B	71.83	68.89	80.71	38.97	32.41	44.45	76.57	32.55	48.49	56.71
RankT5	T5-base	72.13	67.91	75.63	34.99	41.24	42.39	73.37	30.86	44.07	52.19
	T5-large	72.82	67.37	75.45	36.27	39.34	42.90	74.84	32.53	46.81	54.48
	T5-3b	71.09	68.67	80.43	37.43	40.41	42.69	76.58	31.77	48.05	55.91
Inranker	Inranker-small	69.81	61. 68	77.75	35.47	28.83	44.51	74.90	29.37	46.29	50.91
	Inranker-base	71.84	66.30	79.84	36.58	28.97	46.50	76.18	30.46	47.88	54.27
	Inranker-3b	72.71	67.09	81.75	38.25	29.24	47.62	78.31	32.20	49.63	62.47
Transformer Ranker	mxbai-rerank-base	72.49	67.15	84.00	35.64	34.32	42.50	72.33	30.20	51.92	55.59
	bge-reranker-v2-m3	72.19	66.98	74.79	33.84	39.85	41.93	73.48	31.36	45.84	48.44
	jina-reranker-turbo	70.35	63.62	77.97	37.29	30.80	41.75	74.53	28.46	42.79	44.19
Splade Reranker	Splade Cocondenser	71.47	66.18	68.87	34.95	37.96	41.25	68.72	32.27	43.28	47.51
Sentence Transformer	GTR-T5-base	68.09	62.40	70.10	32.02	32.70	36.20	60.23	30.79	43.24	45.38
	GTR-T5-xxl	68.53	64.07	72.70	34.02	36.77	39.90	65.62	31.37	47.01	49.67
colbert ranker	colbert-v2	69.02	66.78	72.6	33.70	35.51	45.20	67.74	33.01	41.21	45.83
monoBERT	BERT (340M)	70.50	67.28	70.01	36.88	31.75	41.87	71.36	31.44	44.62	49.35
Cohere Rerank-v2	-	73.22	67.08	81.81	36.36	32.51	42.51	74.44	29.60	47.59	50.78
Promptagator++	-	-	-	76.2	37.0	38.1	43.4	73.1	-	-	-
TWOLAR	TWOLAR-Large	72.82	67.61	84.30	35.70	33.4	47.8	75.6	33.9	52.7	58.3
	TWOLAR-XL	73.51	70.84	82.70	36.60	37.1	48.0	76.5	33.8	50.8	57.9

Table 5: Performance comparison (nDCG@10) of various pointwise reranking models across standard TREC Deep Learning (DL19, DL20) and multiple BEIR benchmark datasets. First-stage retrieval uses BM25 to retrieve top-100 candidates, which are then re-ranked by each model.

Method	Model	DL19	DL20	Covid	NFC	Touche	DBPedia	SciFact	Signal	News	Robust04
InContext	Mistral-7B	59.2	53.6	63.9	33.2	-	31.4	72.4	-	-	-
	Llama-3.18B	55.7	51.9	72.8	34.7	-	35.3	76.1	-	-	-
RankGPT	Llama-3.2-1B	47.1	44.9	59.2	31.4	37.2	26.04	67.4	26.9	38.7	38.1
	Llama-3.2-3B	58.0	53.3	68.1	31.8	37.2	36.1	67.3	32.7	42.4	44.8
	gpt-3.5-turbo	65.80=	62.9	76.6	35.6	36.1	44.4	70.4	32.1	48.8	50.6
	gpt-4	75.5	70.5	85.5	38.4	38.5	47.1	74.9	34.4	52.8	57.5
	llama 3.1 8b	58.4	59.6	69.6	33.6	37.9	37.2	69.8	32.9	43.9	49.5
ListT5	listt5-base	71.8	68.1	78.3	35.6	33.4	43.7	74.1	33.5	48.5	52.1
	listt5-3b	71.8	69.1	84.7	37.7	33.6	46.2	77.0	33.8	53.2	57.8
lit5dist	LiT5-Distill-base	72.4	67.9	70.4	32.6	33.6	42.7	56.3	34.1	41.5	44.3
	LiT5-Distill-large	73.1	70.3	73.7	34.9	33.4	43.1	66.7	30.8	44.4	52.4
	LiT5-Distill-xl	72.4	72.4	72.9	35.8	32.7	43.5	71.8	31.2	46.5	53.7
	LiT5-Distill-base-v2	71.6	69.1	70.5	34.2	34.2	43.1	67.2	33.2	45.2	48.0
lit5score	LiT5-Score-xl	69.3	65.5	69.6	34.3	29.0	39.1	67.5	24.0	44.9	52.8
Vicuna Reranker	Vicuna 7b	67.1	65.2	78.3	32.9	32.7	43.2	70.4	32.8	44.9	47.8
Zephyr Reranker	Zephyr 7B	74.2	70.2	80.7	36.5	31.1	43.1	75.1	31.9	48.9	54.2

Table 6: Evaluation results (nDCG@10) of listwise reranking approaches on TREC Deep Learning (DL19, DL20) and selected BEIR benchmarks. Models re-rank top-100 documents initially retrieved by BM25.

such as FlashRank (MiniLM-L-12-v2, 41.02% on NQ-Test, 37.05% on WebQ) and MonoBERT-large (39.05% on NQ-Test, 34.99% on WebQ), balance efficiency and effectiveness. Listwise models like Twolar-xl (46.84% on NQ-Test, 41.68% on WebQ) and LiT5-Distill-xl (47.81% on NQ-Test, 42.37% on WebQ) also excel, as does GTR-xxl (42.93% on NQ-Test, 39.41% on WebQ). RANKIFY's auto-indexing with FAISS and API-based embeddings (e.g., GPT) streamlines re-ranking, while online retrieval via Craw4AI expands dataset diversity for future experiments. For the re-ranking experiments on BEIR and TREC Deep Learning benchmarks (Tables 5, 6, 8), we employ a two-stage retrieval pipeline. In the first stage, we use BM25 as the initial retriever to generate a pool of candidate documents. Specifically, we retrieve the top-100 documents for each query using the BM25. These top-100 candidates are then re-ranked using various pointwise, pairwise, and listwise re-ranking models. For evaluation, we report normalized Discounted Cumulative Gain at rank 10 (nDCG@10), which measures the quality of the top-10 documents after re-ranking.

Pointwise Reranking (Table 5). MonoT5-3B emerges as a top-performing pointwise model, achieving nDCG@10 of 71.83 on DL19 and 80.71 on Covid. TWOLAR-XL also demonstrates strong performance with 73.51 on DL19 and 82.70 on Covid. FlashRank's MiniLM model (70.40 on DL19) and ColBERT-v2 (69.02 on DL19) strike a compelling balance between ranking accuracy and efficiency, making them practical for real-time systems. Lighter UPR variants (e.g., T5-Small with 49.94 on DL19) lag significantly behind, indicating limitations in handling diverse retrieval tasks without extensive model capacity. This pattern holds consistently across BEIR benchmarks, with larger models achieving higher nDCG@10 scores across NFCorpus, SciFact, and Robust04 datasets.

Listwise Reranking (Table 6). Listwise models like Zephyr-7B and Vicuna-7B demonstrate strong performance, achieving nDCG@10 of 74.22 and

Reranking/	Model	NQ			WebQ		
		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
	gpt2-xl	27.28	63.24	75.84	23.57	60.48	72.73
	gpt-neo-2.7B	28.75	64.81	76.56	24.75	59.64	72.63
RankGPT	llamav3.1-8b	41.55	66.17	75.42	38.77	62.69	73.12
FlashRank	TinyBERT-L-2-v2	31.49	61.57	74.95	28.54	60.62	73.17
	MultiBERT-L-12	11.99	43.54	69.63	12.54	45.91	67.91
	ce-esci-MiniLM-L12-v2	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	base	43.04	68.47	76.28	36.95	64.27	74.45
	large	45.54	70.02	76.81	38.77	66.48	74.31
	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	Meta-Llama-31-8B	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	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 Reranker	llamav3.1-8b	15.15	57.11	76.48	18.89	52.16	71.70
Lit5	LiT5-Distill-base	40.05	65.95	75.73	36.76	63.48	73.12
	LiT5-Distill-large	44.40	67.59	76.01	39.66	64.56	73.67
	LiT5-Distill-xl	47.81	68.55	76.26	42.37	65.55	73.62
	LiT5-Distill-base-v2	42.57	66.73	75.56	39.61	64.22	73.32
	LiT5-Distill-large-v2	46.53	67.83	75.87	41.97	65.64	72.98
	LiT5-Distill-xl-v2	47.92	69.03	76.17	41.53	65.69	73.27
Sentence Transformer Reranker	GTR-base	39.41	65.95	76.03	36.56	64.32	73.62
	GTR-large	40.63	68.25	76.73	38.97	65.30	73.57
	T5-base	31.19	63.60	76.06	29.77	62.84	73.52
	T5-large	30.80	63.35	76.37	30.51	61.71	73.37
	all-MiniLM-L6-v2	33.35	65.37	76.01	30.95	62.10	73.52
	GTR-xl	41.55	67.78	76.81	38.92	66.04	74.01
	GTR-xxl	42.93	68.55	77.00	39.41	65.89	74.01
	T5-xxl	38.89	67.78	76.64	35.82	65.20	74.01
	Bert-co-condensor	30.96	61.91	75.20	32.43	62.20	73.08
	Roberta-base-v2	32.60	63.24	75.42	31.34	62.64	73.37

Table 7: Performance of re-ranking methods on BM25-retrieved documents for NQ-Test and WebQ-Test. Results report Top-1, Top-5, Top-10, Top-20, and Top-50 accuracy. Note: experiments use top-100 retrieved documents; some original studies (e.g., UPR) used 1,000 documents, which may cause result discrepancies.

Method	Model	DL19	DL20	Covid	NFC	Touche	DBPedia	SciFact	Signal	News	Robust04
PRP	FLAN-T5-XL	68.66	66.59	77.58	-	40.48	44.77	73.43	35.62	46.45	50.74
	FLAN-T5-XXL	67.00	67.35	74.39	-	41.60	42.19	72.46	35.12	47.26	52.38
	FLAN-UL2	72.65	70.46	79.45	-	37.89	46.47	73.33	35.20	49.11	53.43
EchoRank	Flan-T5-Large	60.29	58.24	54.98	30.06	24.93	35.18	65.09	33.86	19.26	21.51
	Flan-T5-xl	59.62	58.98	55.51	30.57	25.24	35.49	69.14	33.11	19.05	21.62

Table 8: Performance outcomes (nDCG@10) of pairwise reranking methods evaluated on TREC DL benchmarks and various BEIR datasets.

67.19 on DL19, respectively, with Zephyr also excelling on Covid (80.70) and Robust04 (54.20). These results validate the strength of generative rerankers that model full document lists. RankGPT variants show competitive performance, with GPT-4 achieving 75.59 on DL19 and llama-3.1-8b reaching 58.46 on DL19. LiT5-Distill models provide solid performance with the large variant achieving 73.18 on DL19, offering a good balance between effectiveness and computational efficiency. InContext rerankers with Llama-3.1-8B achieve 55.7 on DL19, providing moderate gains over baseline approaches.

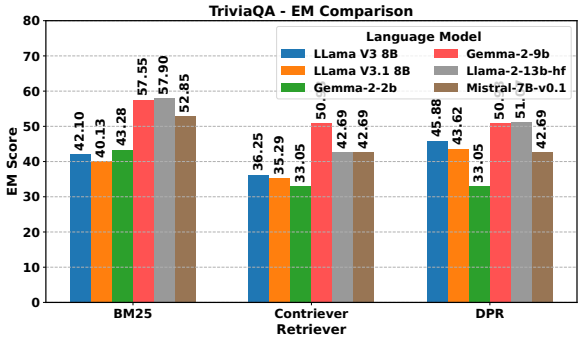

Figure 2: Exact Match (EM) scores on TriviaQA-Test comparing three retrievers (BM25, Contriever, DPR) across six language models (LLaMA V3/V3.1 8B, Gemma-2-2B/9B, LLaMA-2-13B, Mistral-7B). Results for NQ and WebQ datasets are shown in Figures 3 and 4 (Appendix D).

Pairwise Reranking (Table 8). EchoRank variants demonstrate competitive performance, with Flan-T5-xl and Flan-T5-Large reaching nDCG@10 scores of 59.62 and 60.29 on DL19 respectively. PRP models using FLAN-UL2 achieve strong results with 72.65 on DL19 and 79.45 on Covid. These results show that well-optimized pairwise methods can achieve competitive performance on standard IR benchmarks. However, their computational cost—due to pairwise comparisons across document sets—remains a consideration for scalability in production environments.

3.5 End-to-End Pipeline Evaluation

We evaluate RANKIFY's full pipeline capability by combining retrieval, re-ranking, and generation modules. Given the combinatorial space (7 retrievers × 24 re-rankers × 9 RAG methods × 6 LLMs × 42 datasets = 68,000+ configurations), we strategically selected representative combinations across three benchmarks (NQ, TriviaQA, WebQ).

3.5.1 Full Pipeline

Tables 13, 14, and 15 (Appendix C) present comprehensive results combining retrievers (BM25, MSS, Contriever, DPR, MSS+DPR), rerankers (UPR, RankGPT), and generators (LLaMA V3.3/V3.1 8B, Gemma-2-2B/9B, LLaMA-2-13B, Mistral-7B) using in-context learning RAG. We report Exact Match (EM), Recall, and Containment metrics.

Impact of Re-ranking on Generation Quality. Figure 2 demonstrates retrieval-only performance across different retrievers and language models

on TriviaQA. Across all retriever-LLM pairs, re-ranking (UPR, RankGPT) improves EM by 5-15% (detailed results in Tables 13–15, Appendix C). For example, on NQ with LLaMA V3.3 8B: **BM25 alone**: 14.90% EM → **BM25+RankGPT**: 28.45% (+13.55%). **DPR alone**: 28.08% EM → **DPR+RankGPT**: 31.74% (+3.66%).

This demonstrates that re-ranking is essential for generation quality, particularly for weaker initial retrievers. **Dense retrievers benefit more from re-ranking**: DPR+RankGPT achieves 31.74% EM on NQ (LLaMA V3.3 8B), showing dense retrieval + listwise re-ranking synergy. **Model size impacts generation**: Larger LLMs (Gemma-2-9B vs 2B, LLaMA-2-13B vs Mistral-7B) achieve 3-8% improvements across all retriever-reranker combinations. **Retriever quality matters**: DPR and MSS+DPR provide stronger initial context than BM25, leading to better generation even before re-ranking. **Dataset characteristics**: BM25 performs best on TriviaQA (57.55% with Gemma-2-9B and 57.90% with LLaMA-2-13B), while DPR excels on NQ and WebQ (see Figures 3 and 4 in Appendix D).

Model	Dataset	Method	EM	F1	Prec	Rec	Con
LLaMA 3.1-8B	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
LLaMA 3.1-8B	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 9: Comparison of RAG methods including Basic RAG, Chain-of-Thought (CoT), and Zero-Shot generation using LLaMA-3.1-8B on NQ-Test and WebQ-Test.

3.5.2 RAG Method Comparison

Table 9 evaluates RAG strategies (Basic RAG, Chain-of-Thought, Zero-Shot) using LLaMA-3.1-8B with DPR retrieval. Basic RAG achieves highest performance (17.01% EM on NQ, 12.84% on WebQ), demonstrating that direct retrieval-augmented generation is most effective for factual QA. Zero-shot generation without retrieval performs worst, confirming the importance of retrieval augmentation.

3.5.3 Online Retrieval via Craw4AI

Table 10 validates RANKIFY's real-time retrieval on 100 WebQ-Test queries. Craw4AI achieves competitive performance at higher recall (90.0% Top-100, matching BGE), validating viability for production applications despite lower early precision (34.0% Top-1) due to diverse web results versus curated corpora.

Retrieval Method	Top-1	Top-5	Top-10	Top-20	Top-50	Top-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 10: Online retrieval validation using Craw4AI on 100 randomly sampled queries from WebQ-Test. Online retrieval achieves competitive performance, particularly at higher recall levels (90.0% at Top-100).

3.6 Generation Quality Assessment

Table 11 evaluates retrieval-generation pipelines using RAGAS-inspired metrics (Es et al., 2024) (faithfulness, answer relevancy, context precision) on NQ-Test with LLaMA-3.1-8B. Re-ranking consistently improves all metrics: BM25+RankGPT increases faithfulness from 28.7 to 35.8 (+7.1) and answer relevancy from 21.8 to 31.8 (+10.0). DPR+RankGPT achieves best overall performance (29.9 EM, 37.8 faithfulness, 35.3 relevancy, 27.8 context precision). Context precision shows strongest gains (+8.5 for BM25, +7.7 for DPR), indicating re-ranking promotes relevant documents to early positions, reducing hallucination and improving answer grounding.

Retriever	Reranker	EM	Faithfulness	Ans. Rel.	Ctx. Prec.
BM25	None	12.8	28.7	21.8	14.6
BM25	UPR	24.3	34.8	30.9	20.3
BM25	RankGPT	26.7	35.8	31.8	23.1
DPR	None	23.2	32.8	24.4	20.1
DPR	UPR	24.8	35.9	32.2	25.5
DPR	RankGPT	29.9	37.8	35.3	27.8

Table 11: RAGAS-inspired metrics on NQ-Test using LLaMA-3.1-8B. Faithfulness, Answer Relevancy (Ans. Rel.), and Context Precision (Ctx. Prec.) scores range from 0-100.

4 Conclusion

We introduced in this paper RANKIFY, a modular toolkit that unifies retrieval, re-ranking, and retrieval-augmented generation (RAG) within a cohesive framework. By integrating diverse retrievers, state-of-the-art re-rankers, and seamless RAG pipelines, RANKIFY streamlines experimentation and benchmarking in information retrieval. Our experiments demonstrate its effectiveness in enhancing ranking quality and retrieval-based text generation.

Limitations

Despite the broad capabilities and modular design of RANKIFY, several limitations remain that warrant discussion and future attention.

First, while RANKIFY supports a wide range of retrievers and re-rankers, the framework currently emphasizes English-language datasets and models. This limits its applicability for multilingual or cross-lingual retrieval tasks, which are increasingly important in global applications.

Second, although the toolkit supports real-time online retrieval via Craw4AI and precomputed corpora like Wikipedia and MS MARCO, its integration with dynamic, domain-specific corpora (e.g., biomedical, legal, or financial documents) is limited. Incorporating support for customizable indexing pipelines and domain-specific retriever fine-tuning remains an open challenge.

Lastly, although RANKIFY supports 40 pre-retrieved datasets, the effectiveness of its default retriever–re-ranker combinations may vary across datasets due to differences in domain, length, and question complexity. While we provide curated defaults, optimal configuration may still require manual tuning or search over the model space. RANKIFY currently focuses on benchmarking and evaluation rather than model training. This design decision prioritizes standardized evaluation protocols and reproducibility through pre-computed resources. However, we recognize that training support would enable researchers to develop and benchmark custom models within the same framework.

Future versions will incorporate training capabilities for retrievers and re-rankers, potentially through integration with existing training frameworks like Transformers and PyTorch Lightning.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, and 3 others. 2016. Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA. USENIX Association.

Abdelrahman Elsayed Mahmoud Abdallah, Jamshid Mozafari, Bhawna Piryani, Mohammed M.Abdelgwad, and Adam Jatowt. 2025. DynRank: Improve passage retrieval with dynamic zero-shot prompting based on question classification. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4768–4778, Abu Dhabi, UAE. Association for Computational Linguistics.

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*.

Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the yodaqa system. In *International Conference of the cross-language evaluation Forum for European languages*, pages 222–228. Springer.

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.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.

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.

Tao Chen, Mingyang Zhang, Jing Lu, Michael Bendersky, and Marc Najork. 2022. Out-of-domain semantics to the rescue! zero-shot hybrid retrieval models. In *European Conference on Information Retrieval*, pages 95–110. Springer.

Gobinda G Chowdhury. 2010. *Introduction to modern information retrieval*. Facet publishing.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*.

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.

Daniel Cohen, Liu Yang, and W Bruce Croft. 2018. Wikipassageqa: A benchmark collection for research on non-factoid answer passage retrieval. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 1165–1168.

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*.

W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading.

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.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement lexical retrieval model with semantic residual embeddings. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I*, volume 12656 of *Lecture Notes in Computer Science*, pages 146–160. Springer.

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*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

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, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. 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.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, Adriane Boyd, and 1 others. 2020. spacy: Industrial-strength natural language processing in python.

Chao-Wei Huang and Yun-Nung Chen. 2024. Pairdistill: Pairwise relevance distillation for dense retrieval. *arXiv preprint arXiv:2410.01383*.

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.

Ashwin Kalyan, Abhinav Kumar, Arjun Chandrasekaran, Ashish Sabharwal, and Peter Clark. 2021. How much coffee was consumed during emnlp 2019? fermi problems: A new reasoning challenge for ai. *arXiv preprint arXiv:2110.14207*.

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)*.

Danupat Khamnuansin, Tawunrat Chalothorn, and Ekapol Chuangsuwanich. 2024. Mrrank: Improving question answering retrieval system through multi-result ranking model. *arXiv preprint arXiv:2406.05733*.

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.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

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.

Zheng Liu, Yujia Zhou, Yutao Zhu, Jianxun Lian, Chaozhuo Li, Zhicheng Dou, Defu Lian, and Jian-Yun Nie. 2024. Information retrieval meets large language models. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 1586–1589.

Xinwei Long, Jiali Zeng, Fandong Meng, Zhiyuan Ma, Kaiyan Zhang, Bowen Zhou, and Jie Zhou. 2024. Generative multi-modal knowledge retrieval with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18733–18741.

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*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *EMNLP*.

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.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, and 1 others. 2023. Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563*.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

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)*.

Amit Singhal and 1 others. 2001. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43.

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.

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.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019. QuaRTz: An open-domain dataset of qualitative relationship questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5941–5946, Hong Kong, China. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

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*.

Jiajia Wang, Jimmy Xiangji Huang, Xinhui Tu, Junmei Wang, Angela Jennifer Huang, Md Tahmid Rahman Laskar, and Amran Bhuiyan. 2024a. Utilizing bert for information retrieval: Survey, applications, resources, and challenges. *ACM Computing Surveys*, 56(7):1–33.

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.

Shuting Wang, Jiongnan Liu, Shiren Song, Jiehan Cheng, Yuqi Fu, Peidong Guo, Kun Fang, Yutao Zhu, and Zhicheng Dou. 2024b. Domainrag: A chinese benchmark for evaluating domain-specific retrieval-augmented generation. *arXiv preprint arXiv:2406.05654*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Haoyi Xiong, Jiang Bian, Yuchen Li, Xuhong Li, Mengnan Du, Shuaiqiang Wang, Dawei Yin, and Sumi Helal. 2024. When search engine services meet large language models: visions and challenges. *IEEE Transactions on Services Computing*.

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)*.

Liu Yang, Junjie Hu, Minghui Qiu, Chen Qu, Jianfeng Gao, W Bruce Croft, Xiaodong Liu, Yelong Shen, and Jingjing Liu. 2019. A hybrid retrieval-generation neural conversation model. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1341–1350.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

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.

Soyoung Yoon, Eunbi Choi, Jiyeon Kim, Hyeongu Yun, Yireun Kim, and Seung won Hwang. 2024. Listt5: Listwise reranking with fusion-in-decoder improves zero-shot retrieval. *Preprint*, arXiv:2402.15838.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

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 Related Work

Information retrieval (IR) has evolved from lexical models like BM25 (Robertson et al., 1995) to dense retrievers (e.g., DPR (Yates et al., 2021), ANCE (Xiong et al., 2021), ColBERT (Khattab and Zaharia, 2020)) using neural encoders for semantic matching. Hybrid retrievers (Gao et al., 2021) combine lexical and semantic features, with advancements like knowledge distillation (Qu et al., 2021) improving performance. Re-ranking has progressed from pointwise (Zhu et al., 2021) and pairwise (Cao et al., 2007) to listwise methods like LambdaRank (Burges, 2010) and RankT5 (Zhuang et al., 2023). Deep neural models and LLMs (e.g., GPT-4 (Achiam et al., 2023)) enable zero-shot re-ranking, capturing complex query-document interactions.

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) enhances LLMs for knowledge-intensive tasks by integrating retrieved documents, improving accuracy (Wang et al., 2022b). Its effectiveness depends on high-quality retrieval and re-ranking, necessitating robust frameworks.

Frameworks like Rerankers (Clavié, 2024) and RankLLM focus on re-ranking, while FlashRAG (Jin et al., 2024) and AutoRAG (Kim et al., 2024) support RAG but lack dataset diversity. LangChain (Chase, 2022) and DSPy (Khattab et al., 2023) simplify workflows. Table 1 compares these, highlighting RANKIFY 's strengths in dataset and model support.

B RANKIFY Dataset

This section provides a comprehensive overview of all 42 datasets supported in RANKIFY, organized by task category. Our dataset collection spans diverse information retrieval and natural language processing tasks, enabling systematic evaluation across different retrieval, re-ranking, and generation scenarios.

B.1 Dataset Organization and Coverage

RANKIFY's dataset collection is organized into 9 primary task categories, each representing distinct challenges in information retrieval and question answering:

Question Answering (QA): The largest category with 17 datasets covering diverse QA paradigms including factoid QA (NQ, TriviaQA), reading comprehension (SQuAD), entity-centric questions (WebQ, PopQA), commonsense reasoning (CommonsenseQA, PIQA), and narrative understanding (NarrativeQA). These datasets range from 2,290 examples (StrategyQA) to over 800,000 examples (MSMARCO-QA), providing comprehensive coverage of single-hop question answering scenarios.

Multi-Hop Question Answering: Four datasets (2WikiMultiHopQA, HotpotQA, Musique, Bamboogle) specifically designed to evaluate reasoning across multiple retrieved documents. These datasets require systems to integrate information from 2-4 supporting passages, testing the limits of retrieval precision and reasoning capabilities.

Temporal QA: Two large-scale datasets (ArchivalQA with 481K examples, ChroniclingAmericaQA with 429K examples) focusing on temporally-grounded questions where retrieval must account for time-sensitive information and historical context.

Long-Form QA: Datasets (ELI5, ASQA) requiring paragraph-length answers rather than short factoid responses, evaluating generation quality and informativeness beyond simple answer extraction.

Multiple-Choice: Six datasets (MMLU, ARC, OpenBookQA, etc.) spanning diverse domains from scientific reasoning to truthfulness assessment, with answer candidates provided for each question.

Entity Linking: Three datasets (WNED, AIDA CoNLL-YAGO, EntityQuestions) for evaluating entity recognition and disambiguation in retrieved contexts.

Slot Filling: Two relation extraction datasets (Zero-shot RE, T-REx) with over 2.4 million examples for evaluating structured knowledge extraction.

Specialized Tasks: Individual datasets for dialog generation (WOW), fact verification (FEVER), and open-domain summarization (WikiAsp), plus IR-specific benchmarks (BEIR, DL19/20) and domain-specific evaluation (DomainRAG).

B.2 Dataset Preparation and Preprocessing

All datasets in RANKIFY undergo standardized preprocessing to ensure consistency:

- **Schema Standardization:** Each dataset is converted to a uniform format with `Question`, `Answers`, and `Contexts` fields
- **Pre-retrieved Documents:** 1,000 top-ranked documents per query retrieved using BM25, DPR, and ColBERT from Wikipedia (December 2018 dump) or MS MARCO corpus
- **Pre-built Indices:** FAISS indices and API-based embeddings (e.g., GPT) pre-computed for all retrievers to eliminate indexing overhead
- **Quality Control:** Answer normalization, duplicate removal, and consistency validation across splits

B.3 Access and Usage

All datasets are hosted on Hugging Face (https://huggingface.co/datasets/abdoelsayed/reranking-datasets-light) and can be downloaded automatically or cached locally. Custom datasets can be created using the `Dataset` class with the same standardized schema.

B.4 Dataset Statistics

Table 12 provides detailed statistics for each dataset, including the number of training, validation, and test examples. Key characteristics:

- **Total examples:** Over 6 million questions across all datasets
- **Corpus coverage:** 21M Wikipedia passages, 8.8M MS MARCO passages
- **Task diversity:** 9 distinct task types covering major IR/NLP evaluation scenarios
- **Domain coverage:** General knowledge, science, history, commonsense, entities, temporal reasoning
- **Split availability:** 30 datasets with train/dev/test splits, 12 test-only datasets

This comprehensive dataset collection enables systematic benchmarking across diverse retrieval scenarios, supporting robust evaluation of retrieval, re-ranking, and RAG methods.

C Multi-Module Pipeline Results

This section presents comprehensive results for all retriever-reranker-generator combinations evaluated in Section 3.5. We report Exact Match (EM), Recall (Rec), and Containment (Con) metrics across three datasets (NQ, TriviaQA, WebQ) for six different language models. These results demonstrate RANKIFY's flexibility in supporting diverse module combinations and validate the synergistic effects of combining strong retrievers with advanced re-ranking methods.

C.1 Experimental Configuration

All experiments in this section follow a consistent setup to ensure fair comparison:

Retrieval Configuration:

- **Retrievers tested:** BM25 (sparse baseline), MSS (Maximum Sparse Score), Contriever (unsupervised dense), DPR (supervised dense), MSS+DPR (hybrid)
- **Corpus:** English Wikipedia (December 2018, 21M passages)
- **Retrieved documents:** Top-100 passages per query

Re-ranking Configuration:

- **Baseline (R):** Retrieval-only, no re-ranking applied
- **UPR:** Unsupervised Passage Re-ranking with T5-base encoder
- **RankGPT:** Listwise re-ranking using LLaMA-3.1-8B
- **Re-ranking pool:** Top-100 retrieved documents

Generation Configuration:

Task	Dataset Name	# Train	# Val	# Test
QA	NQ (Kwiatkowski et al., 2019)	79,168	8,757	3,610
	TriviaQA (Joshi et al., 2017)	78,785	8,837	11,313
	WebQ (Berant et al., 2013)	3,778	-	2,032
	SQuAD (Rajpurkar et al., 2016)	87,599	10,570	-
	NarrativeQA (Kočiskỳ et al., 2018)	32,747	3,461	10,557
	MSMARCO-QA (Nguyen et al., 2017)	808,731	101,093	-
	PopQA (Mallen et al., 2022)	-	-	14,267
	SIQA (Sap et al., 2019)	33,410	1,954	-
	Fermi (Kalyan et al., 2021)	8,000	1,000	1,000
	WikiQA (Yang et al., 2015)	20,360	2,733	6,165
	AmbigQA (Min et al., 2020)	10,036	2,002	-
	CommenseQA (Talmor et al., 2019)	9,741	1,221	-
	PIQA (Bisk et al., 2019)	16,113	1,838	-
	BoolQ (Clark et al., 2019)	9,427	3,270	-
	StrategyQA (Geva et al., 2021)	2,290	-	-
	CuratedTREC (Baudiš and Šedivỳ, 2015)	430	-	430
	WikiPassageQA (Cohen et al., 2018)	3,332	416	416
Multi-Hop QA	2WikiMultiHopQA (Ho et al., 2020)	15,000	12,576	-
	Bamboogle (Press et al., 2023)	-	-	125
	Musique (Trivedi et al., 2022)	19,938	2,417	-
	HotpotQA (Yang et al., 2018)	90,447	7,405	-
Temp QA	ArchivalQA (Wang et al., 2022a)	384,426	48,304	48,760
	ChroniclingQA (Piryani et al., 2024)	385,629	21,739	21,735
Long-Form QA	ELI5 (Fan et al., 2019)	272,634	1,507	-
	ASQA (Stelmakh et al., 2022)	4,353	948	-
Multiple-Choice	MMLU (Hendrycks et al., 2021b,a)	99,842	1,531	14,042
	TruthfulQA (Lin et al., 2022)	-	817	-
	HellaSwag (Zellers et al., 2019)	39,905	10,042	-
	ARC (Clark et al., 2018)	3,370	869	3,548
	OpenBookQA (Mihaylov et al., 2018)	4,957	500	500
	QuaRTz (Tafjord et al., 2019)	2,696	384	784
Entity-linking	WNED (Tedeschi et al., 2021)	-	8,995	-
	AIDA CoNLL-YAGO (Hoffart et al., 2011)	18,395	4,784	-
	EntityQuestions (Sciavolino et al., 2021)	-	-	2,472
Slot filling	Zero-shot RE (Levy et al., 2017)	147,909	3,724	-
	T-REx (ElSahar et al., 2018)	2,284,168	5,000	-
Dialog Generation	WOW (Dinan et al., 2019)	63,734	3,054	-
Fact Verification	FEVER (Thorne et al., 2018)	104,966	10,444	-
Open-domain Summarization	WikiAsp (Hayashi et al., 2020)	300,636	37,046	37,368
IR	BEIR (Thakur et al., 2021)	-	-	6,980
	DL19/20 (Craswell et al., 2020)	-	-	43/54
Specialized	DomainRAG (Wang et al., 2024b)	-	-	485

Table 12: Summary of datasets used in RANKIFY. Categories include QA, Multi-Hop QA, Long-Form QA, Multiple-Choice, Entity-Linking, Slot Filling, Fact Verification, Dialog Generation, Summarization, IR benchmarking, and other specialized datasets. Total: 42 datasets.

- **RAG method:** In-context learning with top-1 re-ranked passages
- **Context window:** Maximum 2,048 tokens
- **Generation parameters:** Temperature=0.1, max_new_tokens=100
- **Prompt format:** Question + concatenated contexts + instruction

Hardware: All experiments conducted on 2× NVIDIA A100 GPUs (80GB) with consistent batch sizes and inference settings across all model combinations.

C.2 Key Findings Across All Experiments

Analysis of Tables 13–15 reveals several critical patterns:

1. Re-ranking Impact on Generation Quality:

- Re-ranking consistently improves EM scores by 5-15% across all retriever-LLM pairs
- Largest improvements observed for weaker retrievers (BM25: +13.55% on NQ with LLaMA V3.3 8B)
- Dense retrievers (DPR, MSS+DPR) show smaller but consistent gains (+3-7%)
- UPR provides stable improvements (8-12%) across all configurations
- RankGPT achieves highest absolute performance but with higher computational cost

2. Retriever Quality Hierarchy:

- DPR and MSS+DPR consistently outperform sparse methods (BM25, MSS) on factoid QA (NQ, WebQ)
- BM25 shows competitive performance on TriviaQA (42-58% EM), indicating strong lexical overlap
- Contriever provides intermediate performance, positioned between sparse and supervised dense methods
- Hybrid approaches (MSS+DPR) achieve best balance of precision and recall

3. Model Size and Capacity Effects:

- Larger models benefit more from high-quality retrieval: LLaMA-2-13B shows 7-10% EM improvement with DPR vs BM25
- Smaller models (Mistral-7B) show lower absolute performance (6-25% EM) but similar relative improvements from re-ranking
- Gemma-2-9B vs 2B comparison demonstrates consistent 3-8% improvements from model scaling
- Model architecture matters: LLaMA V3.3 outperforms V3.1 by 1-3% across identical configurations

4. Dataset-Specific Patterns:

- TriviaQA: Highest overall EM (42-62%), benefits less from re-ranking due to strong lexical signals
- NQ: Moderate EM (12-33%), shows largest re-ranking improvements, benefits from dense retrieval
- WebQ: Lowest EM (6-22%), entity-centric queries require both strong retrieval and re-ranking

C.3 LLaMA V3.3 8B and LLaMA V3.1 8B Results

Table 13 compares the latest LLaMA V3.3 8B model against its predecessor V3.1 8B across all retriever-reranker combinations. Key observations:

Performance Improvements in V3.3:

- Average +2.1% EM improvement over V3.1 across all configurations
- Largest gains on NQ (+2.5% average) and WebQ (+1.8% average)
- Consistent improvements across all retrievers, suggesting better instruction following and reasoning
- Re-ranking benefits preserved: both versions show similar relative improvements from UPR/RankGPT

Best Configurations:

- **NQ:** DPR+RankGPT achieves 31.74% EM (V3.3) vs 29.58% (V3.1)
- **TriviaQA:** BM25+UPR achieves 55.85% EM (V3.3) vs 55.22% (V3.1)
- **WebQ:** MSS+DPR+RankGPT achieves 21.41% EM (V3.3) vs 18.99% (V3.1)

The results validate that architectural improvements in newer LLMs transfer effectively to RAG scenarios, with benefits compounding when combined with strong retrieval and re-ranking.

C.4 Gemma-2-2B and Gemma-2-9B Results

Table 14 examines the impact of model scaling within the Gemma-2 family. The 4.5× parameter increase (2B → 9B) reveals interesting scaling behaviors:

Scaling Effects:

- **NQ:** +5.8% average EM improvement (2B: 14-34% → 9B: 20-34%)
- **TriviaQA:** +14.3% average improvement (2B: 33-59% → 9B: 51-58%)
- **WebQ:** Minimal improvement (+0.3%), suggesting entity-centric queries are bottlenecked by retrieval quality

Robustness Observations:

- Gemma-2-2B shows remarkably consistent performance across different retrievers (14.71-14.96% on WebQ), indicating strong robustness to context quality variations
- Gemma-2-9B benefits more from re-ranking on complex datasets (TriviaQA: +15.5% with UPR vs +15.4% for 2B)
- Both models achieve peak performance with DPR+RankGPT: 34.16% (2B) and 34.04% (9B) on NQ

These results suggest that mid-sized models (2-9B parameters) can achieve competitive RAG performance when paired with strong retrieval, making them viable for resource-constrained deployments.

C.5 LLaMA-2-13B and Mistral-7B Results

Table 15 contrasts a large model (LLaMA-2-13B) with a smaller, efficient alternative (Mistral-7B-v0.1):

Performance Gap Analysis:

- **NQ:** LLaMA-2-13B outperforms Mistral-7B by +10.0% average EM (21-33% vs 11-25%)
- **TriviaQA:** +15.2% gap (42-62% vs 43-60%), with LLaMA-2-13B achieving highest overall EM (61.60% with UPR)
- **WebQ:** +13.4% gap (20-22% vs 6-17%), highlighting Mistral's challenges with entity-centric queries

Re-ranking Effectiveness:

- LLaMA-2-13B shows consistent 6-8% improvements from UPR across all datasets
- Mistral-7B exhibits larger relative improvements (+14.0% on NQ with UPR), indicating it benefits more from better context ordering
- RankGPT provides additional +1-5% over UPR for both models

Computational Trade-offs: While LLaMA-2-13B achieves superior performance, its 1.9× parameter increase over Mistral-7B translates to approximately 2.3× inference latency. For applications requiring real-time responses, Mistral-7B with strong re-ranking (achieving 25.35% EM on NQ) may offer better latency-accuracy trade-offs than LLaMA-2-13B without re-ranking (21.94% EM).

D Additional Generator Results Across Datasets

This section presents complementary results for the retrieval-only baseline experiments discussed in Section 5.4. While Figure 2 in the main text focuses on TriviaQA results, Figures 3 and 4 below show the complete performance comparison across Natural Questions (NQ) and WebQuestions (WebQ) datasets.

D.1 Natural Questions Results

Key observations for NQ:

- DPR consistently achieves highest EM scores across all language models, demonstrating its effectiveness for factoid QA
- LLaMA V3.3 8B with DPR achieves 28.08% EM, significantly outperforming BM25 (14.90%) by +13.18%
- Contriever shows intermediate performance (15.29% with LLaMA V3.3 8B), positioned between BM25 and DPR
- Larger models benefit more from dense retrieval: the gap between DPR and BM25 increases with model size
- Mistral-7B shows notably lower performance (11.19% with BM25), suggesting it may require different prompting strategies

		LLaMA V3.3 8B									LLaMA V3.1 8B								
		NQ			TriviaQA			WebQ			NQ			TriviaQA			WebQ		
Retriever	**Re-rank**	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con
BM25	R	14.90	26.68	19.91	42.10	56.90	50.11	10.23	24.95	16.92	12.82	27.14	20.27	40.13	58.40	51.40	9.25	24.92	17.22
	UPR	23.49	41.02	31.55	55.85	69.59	62.65	17.27	44.15	32.68	24.24	40.68	31.47	55.22	68.14	61.78	17.67	41.68	30.71
	RankGPT	28.45	42.73	34.02	-	-	-	19.73	40.57	29.82	26.65	40.89	32.44	-	-	-	18.55	37.92	28.10
MSS	R	12.82	22.96	17.36	31.90	44.27	37.75	7.38	18.90	11.81	11.19	23.37	17.81	30.74	45.82	38.91	6.88	19.30	12.40
	UPR	23.35	40.52	31.55	53.78	68.67	61.63	17.08	43.97	32.14	24.43	41.08	31.77	54.97	68.46	61.73	16.70	42.20	31.30
	RankGPT	28.17	41.67	33.21	-	-	-	19.05	39.33	29.58	25.84	39.07	31.41	-	-	-	17.37	36.38	27.21
Contriever	R	15.29	27.36	20.99	36.25	49.52	49.52	10.67	28.28	20.22	13.24	27.68	21.88	35.29	50.96	44.19	9.35	28.71	20.32
	UPR	23.24	41.01	31.61	53.48	68.70	61.82	17.32	44.11	32.97	24.21	40.99	31.63	55.71	68.74	61.98	17.62	37.73	28.44
	RankGPT	30.55	44.25	35.40	-	-	-	19.78	41.18	31.10	28.86	42.42	33.68	-	-	-	17.62	37.73	28.44
DPR	R	28.08	45.40	36.37	45.88	61.24	54.58	19.83	40.27	30.98	23.21	44.99	36.03	43.62	62.61	55.97	14.32	38.27	28.98
	UPR	23.60	41.18	31.77	53.41	68.60	61.60	18.06	44.01	32.48	24.74	41.32	31.75	56.07	69.14	62.41	19.64	39.77	29.87
	RankGPT	31.74	46.41	36.76	-	-	-	20.42	40.84	31.00	29.58	43.92	34.65	-	-	-	19.64	39.77	29.87
MSS+DPR	R	28.17	46.72	37.00	47.69	63.66	57.08	13.92	40.87	30.57	23.68	46.70	37.53	45.72	65.24	58.53	13.92	38.97	29.67
	UPR	23.10	40.81	31.25	53.65	68.61	61.69	16.68	44.07	32.48	25.24	41.17	31.83	55.52	68.76	61.90	18.99	40.76	30.22
	RankGPT	31.94	45.95	36.59	-	-	-	21.41	42.54	32.14	29.95	45.00	36.49	-	-	-	18.99	40.76	30.22

Table 13: Multi-module evaluation combining retrievers (BM25, MSS, Contriever, DPR, MSS+DPR), re-rankers (R=Retrieval-only, UPR, RankGPT), and generators (LLaMA V3.3 8B, LLaMA V3.1 8B) using in-context learning RAG on NQ, TriviaQA, and WebQ test sets. Metrics: EM (Exact Match), Rec (Recall), Con (Containment).

		Gemma-2-2B									Gemma-2-9B								
		NQ			TriviaQA			WebQ			NQ			TriviaQA			WebQ		
Retriever	**Re-rank**	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con
BM25	R	14.02	25.55	18.53	43.28	52.78	47.01	14.71	37.74	27.21	19.81	26.95	22.05	57.55	65.53	60.29	14.96	24.87	20.13
	UPR	26.23	44.37	33.99	58.71	71.59	64.79	19.78	48.16	36.47	23.41	41.52	32.24	58.74	71.92	65.20	15.94	46.16	34.15
	RankGPT	30.36	45.96	36.09	-	-	-	21.11	45.13	34.35	30.75	43.48	35.46	-	-	-	21.06	39.32	31.15
MSS	R	13.96	25.41	18.50	33.05	42.24	36.23	14.71	37.74	27.21	19.78	26.86	22.08	50.93	58.57	53.28	14.96	24.92	20.18
	UPR	26.20	44.14	33.77	58.15	71.02	64.10	19.73	48.92	36.96	23.10	41.49	32.05	57.50	71.15	64.33	15.85	47.20	34.65
	RankGPT	29.17	45.00	35.32	-	-	-	19.93	44.61	34.10	29.97	42.59	34.85	-	-	-	19.64	36.91	29.72
Contriever	R	13.96	25.41	18.50	33.05	42.24	36.23	14.71	37.74	27.21	19.78	26.86	22.08	50.93	58.57	53.28	14.96	24.92	20.18
	UPR	26.26	44.26	33.82	58.37	71.17	64.23	19.83	48.32	36.61	23.21	41.46	32.08	57.64	71.20	64.36	16.14	46.12	34.01
	RankGPT	32.11	47.86	38.31	-	-	-	20.67	45.77	34.89	32.44	44.83	36.79	-	-	-	19.88	38.57	30.02
DPR	R	13.99	25.44	18.53	33.05	42.24	36.23	14.71	37.74	27.21	19.78	26.86	22.08	50.93	58.57	53.28	14.96	24.92	20.18
	UPR	26.51	44.62	34.16	58.71	71.72	64.78	19.78	48.62	36.86	23.38	41.80	32.38	57.85	71.54	64.75	16.04	46.56	34.50
	RankGPT	34.16	50.26	39.42	-	-	-	21.21	46.41	35.48	34.04	47.14	38.64	-	-	-	21.01	40.41	31.69
MSS+DPR	R	13.96	25.41	18.50	33.05	42.24	36.23	14.71	37.74	27.21	19.78	26.86	22.08	50.93	58.57	53.28	14.96	24.92	20.18
	UPR	26.54	44.67	34.21	58.64	71.50	64.55	19.49	48.03	36.02	23.46	41.75	32.38	57.75	71.34	64.48	15.94	45.96	33.56
	RankGPT	32.51	49.77	39.53	-	-	-	21.41	47.58	36.52	32.13	46.55	38.29	-	-	-	21.06	40.72	32.04

Table 14: Multi-module evaluation using Gemma-2-2B and Gemma-2-9B as generators with various retriever-reranker combinations. The larger Gemma-2-9B model consistently achieves higher EM scores across all configurations, demonstrating the impact of model capacity on generation quality.

D.2 WebQuestions Results

Key observations for WebQ:

- WebQ shows the largest performance variance across retrievers, reflecting its complex entity-centric queries
- DPR provides consistent improvements over BM25 (average +7.5% across all models)
- Gemma-2-2B and Gemma-2-9B show remarkably similar performance (14.71-14.96% range), suggesting robustness
- LLaMA-2-13B achieves competitive results (19.54-19.98% EM) with relatively small variance across retrievers
- Mistral-7B shows lowest performance (6.40% across all retrievers), indicating potential limitations for entity-focused QA

D.3 Cross-Dataset Analysis

Comparing performance across all three datasets (TriviaQA in main text Figure 2, NQ in Figure 3, WebQ in Figure 4) reveals several important patterns:

1. **Dataset difficulty hierarchy:** TriviaQA (42.10-57.90% EM) > NQ (13.99-28.08% EM) > WebQ (6.40-19.83% EM), reflecting different question complexities and answer distributions
2. **Retriever effectiveness varies by dataset:**
   - TriviaQA: BM25 competitive with dense methods (42.10% vs 45.88% DPR)

		LLaMA-2-13B									Mistral-7B-v0.1								
		NQ			TriviaQA			WebQ			NQ			TriviaQA			WebQ		
Retriever	**Re-rank**	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con	EM	Rec	Con
BM25	R	21.14	30.82	24.46	57.90	65.27	59.57	19.54	37.38	27.51	11.19	13.45	11.80	52.85	58.11	53.82	6.40	8.46	6.84
	UPR	27.59	42.65	32.99	61.60	71.62	65.00	20.37	44.09	33.76	25.18	40.47	31.11	59.64	69.91	63.20	17.18	40.89	30.46
	RankGPT	29.22	43.00	34.21	-	-	-	21.99	41.25	31.15	25.35	40.47	31.36	-	-	-	17.18	40.90	30.46
MSS	R	21.52	30.92	24.18	51.75	58.58	53.35	20.62	39.45	29.68	11.08	13.33	11.66	42.69	47.12	43.40	6.40	8.46	6.84
	UPR	27.29	42.28	32.63	59.23	69.93	63.56	20.72	44.50	33.96	23.77	40.41	30.97	57.24	67.67	61.04	17.42	42.17	31.20
	RankGPT	27.56	41.77	33.05	-	-	-	20.77	44.50	33.96	23.77	39.08	29.28	-	-	-	16.98	39.87	29.43
Contriever	R	20.47	30.13	23.85	42.69	47.12	43.40	19.98	37.74	27.61	11.08	13.33	11.66	42.69	47.12	43.40	6.40	8.46	6.84
	UPR	26.57	41.99	32.44	59.60	70.02	63.58	20.72	44.65	33.56	25.10	40.43	31.02	57.26	67.63	61.11	17.27	40.48	29.97
	RankGPT	30.39	44.60	36.09	-	-	-	20.72	44.65	33.56	25.10	40.48	31.02	-	-	-	17.27	40.48	29.97
DPR	R	21.94	31.36	24.88	51.07	57.97	52.71	19.83	37.35	28.05	11.11	13.36	11.69	42.69	47.12	43.40	6.40	8.46	6.84
	UPR	27.45	42.93	33.02	60.05	70.49	63.96	20.62	44.81	33.91	25.26	40.71	31.36	57.53	67.80	61.31	17.18	41.20	30.66
	RankGPT	32.77	47.56	38.06	-	-	-	22.15	44.37	33.81	25.26	40.71	31.36	-	-	-	17.18	41.20	30.66
MSS+DPR	R	21.47	31.26	24.60	51.35	58.26	53.01	19.83	37.37	27.61	11.08	13.33	11.66	42.69	47.12	43.40	6.40	8.46	6.84
	UPR	27.45	42.91	33.10	60.00	71.52	64.07	19.98	44.81	33.32	25.24	40.59	31.25	57.47	67.80	61.12	16.09	44.33	33.32
	RankGPT	32.61	48.90	40.19	-	-	-	21.95	43.73	33.32	25.24	40.59	31.25	-	-	-	16.09	44.33	33.32

Table 15: Multi-module evaluation using LLaMA-2-13B and Mistral-7B-v0.1 as generators. LLaMA-2-13B demonstrates superior performance across all metrics, achieving up to 32.77% EM on NQ with DPR+RankGPT, highlighting the importance of model selection in the full pipeline.

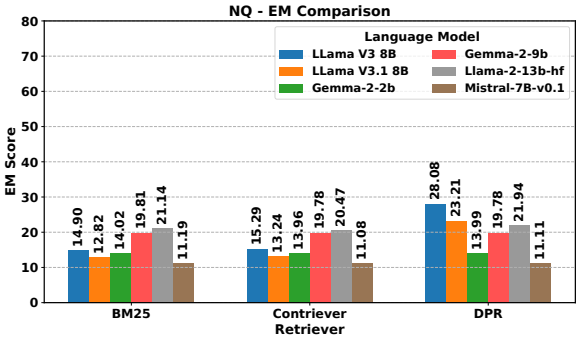

Figure 3: Exact Match (EM) scores on NQ-Test comparing three retrievers (BM25, Contriever, DPR) across six language models. DPR significantly outperforms sparse methods (BM25) and Contriever across all models, with LLaMA V3.3 8B achieving 28.08% EM and LLaMA-2-13B reaching 21.94% EM. The performance gap between DPR and BM25 ranges from 8.24% to 13.18% depending on the model, highlighting the importance of dense retrieval for factoid question answering.

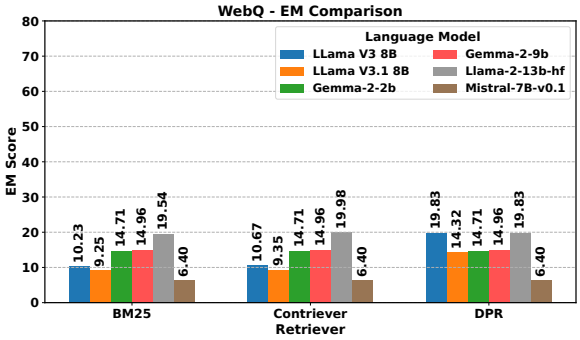

Figure 4: Exact Match (EM) scores on WebQ-Test comparing three retrievers across six language models. WebQ exhibits the most pronounced retriever differences, with DPR achieving 19.83% EM (LLaMA V3.3 8B) compared to 10.23% for BM25 (+9.60%). Gemma-2 models show relatively stable performance across retrievers, while LLaMA-2-13B demonstrates strong results (19.54-19.98% EM range), indicating robustness to retrieval quality variations.

- NQ: DPR shows clear advantage (28.08% vs 14.90% BM25)
- WebQ: DPR provides largest improvements (19.83% vs 10.23% BM25)

3. **Model-dataset interactions:**
   - LLaMA-2-13B excels on TriviaQA (57.90% EM) but shows moderate NQ performance (21.94%)
   - Gemma-2-9B achieves strong TriviaQA results (57.55%) with stable cross-dataset performance
   - LLaMA V3.3 8B provides best NQ performance (28.08%) while maintaining competitive TriviaQA scores
4. **Re-ranking potential:** Datasets with larger BM25-DPR gaps (NQ: 13.18%, WebQ: 9.60%) show greater potential for re-ranking improvements, as confirmed by our full pipeline experiments (Tables 13–15)

These comprehensive results demonstrate that optimal retriever-model combinations are dataset-dependent, reinforcing the need for flexible, modular frameworks like RANKIFY that enable systematic experimentation across different pipeline configurations.

E Generator Architecture

This section outlines the internal modular architecture of the `Generator` module in RANKIFY, as illustrated in Figure 5. The design separates the core RAG methodology from the underlying model backends, enabling high composability and flexibility across retrieval-augmented generation workflows.

Generator Entry Point At the highest level, the `Generator` class acts as a facade that unifies the interaction between users and specific RAG strategies. Upon initialization via `__init__(method: str, model_name: str, backend: str, **kwargs)`, the generator dynamically configures a `BaseRAGMethod` using the provided generation strategy (e.g., `in-context-ralm`, `fid`, `basic`, `cot`). The main method `generate(documents: List[Document])` dispatches document batches to the selected RAG module.

RAG Strategy Modules All RAG methods subclass from `BaseRAGMethod`, which defines a standard interface `answer_questions(documents: List[Document])`. This promotes consistency across implementations and simplifies downstream calls.

The currently supported strategies include:

- **BasicRAG**: A baseline RAG strategy using raw prompt concatenation and direct model invocation.
- **ChainOfThoughtRAG**: A prompting method that guides the model through intermediate reasoning steps.
- **FiDRAGMethod**: Implements Fusion-in-Decoder-style generation by aggregating multiple retrieved contexts into a decoder input.
- **InContextRALMRAG**: A variant of RALM that inserts the retrieved contexts into structured few-shot exemplars for in-context learning.

Each of these modules contains a reference to a `BaseRAGModel` instance for backend-agnostic inference.

Backend-Agnostic RAG Models The `BaseRAGModel` defines two core abstract methods: `generate(prompt: str)` and `embed(text: str)`, which all backends must implement. Four concrete subclasses extend this interface:

- **OpenAIModel**: Wraps OpenAI's API via a `client` and supports LLM-based generation using registered model names.
- **HuggingFaceModel**: Uses Transformers and tokenizers from Hugging Face (`AutoModelForCausalLM`) for local or offline inference.
- **VLLMModel**: Integrates with `vLLM`, enabling high-throughput batched generation on GPU clusters.
- **LiteLLMModel**: Offers lightweight HTTP-based inference using LiteLLM-compatible endpoints.

All model backends include a `PromptGenerator` responsible for formatting the input in a strategy-aware manner (e.g., structuring few-shot examples, separators, or instructions).

PromptGenerator Module Each backend model incorporates a `PromptGenerator` component, which is parameterized by `model_type` and `method`. This component provides the method `generate_system_prompt()` that assembles a system prompt tailored to the RAG strategy and model architecture. This abstraction ensures that prompt formatting is decoupled from model inference.

The instantiation of backends is controlled via the `ModelFactory`, which constructs and returns a `BaseRAGModel` given the target `model_name`, `backend`, and `method`. This factory design encapsulates backend-specific logic and allows dynamic instantiation at runtime.

Summary In summary, the Generator architecture (Figure 5) reflects a clean separation between RAG methods, inference backends, and prompt construction, ensuring high extensibility. New generation strategies or backends can be integrated with minimal refactoring, making RANKIFY suitable for both research exploration and production-scale deployment of retrieval-augmented systems.

F RANKIFY Pipeline

This section illustrates the full architecture of RANKIFY, emphasizing its modular and extensible design. Figure 6 provides a high-level overview of how the system supports both pre-retrieved datasets

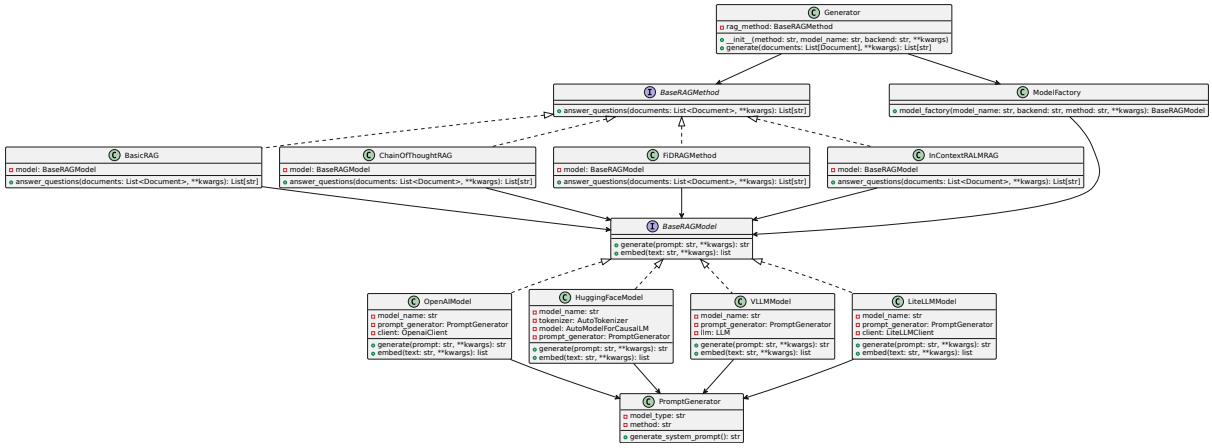

Figure 5: Architecture of the Generator module in RANKIFY, illustrating a modular design that supports pluggable RAG strategies and interchangeable LLM backends.

and real-time document retrieval. Users can interact with the system through components like Craw4AI for online queries or leverage precomputed indices such as FAISS for efficient offline retrieval. The pipeline then proceeds through various re-ranking strategies (pointwise, pairwise, listwise) and concludes with advanced RAG techniques (e.g., Fusion-in-Decoder, In-Context Learning) for final answer generation.

G How to Use RANKIFY

This section offers a practical guide to interacting with the RANKIFY toolkit through concise Python APIs. Each code listing demonstrates a modular aspect of the system's functionality, allowing users to handle dataset inspection, retrieval, re-ranking, and generation in a seamless workflow. The following subsections outline the purpose of each snippet.

Listing 1 demonstrates how to query and display all datasets supported by RANKIFY. The avaiable_dataset() method queries the internal dataset registry and prints metadata (e.g., name, size, source) for each entry. This enables users to quickly identify which benchmark corpora (e.g., NQ, TriviaQA, BEIR) are accessible within the current version of the toolkit.

```
from rankify.dataset.dataset import Dataset
Dataset.avaiable_dataset()
```

Listing 1: This script retrieves the latest information about the available datasets, and displays metadata for each dataset in the terminal.

Listing 2 illustrates how to load a preprocessed dataset from the RANKIFY repository. The user specifies the retriever (e.g., "bm25") and dataset name (e.g., "nq-test"), and optionally disables re-downloading if cached locally. This example loads BM25-retrieved top documents for Natural Questions (NQ), supporting efficient evaluation or re-ranking without re-running initial retrieval.

```
from rankify.dataset.dataset import Dataset
# Load the nq-test dataset retrieved with BM25
dataset = Dataset(retriever="bm25", dataset_name
    ="nq-test")
documents = dataset.download(force_download=
    False)
```

Listing 2: Loading a dataset in Rankify and inspecting its structure.

Listing 3 shows how to programmatically create a custom retrieval dataset from scratch. A Document object is constructed using a Question, Answer, and a list of Context entries with metadata such as confidence score and answer presence. This supports use cases like defining task-specific datasets, bootstrapping from user logs, or generating adversarial benchmarks. The final dataset is persisted to disk via save_dataset().

```
from rankify.dataset.dataset import Dataset,
    Document, Question, Answer, Context
question = Question("What is the capital of
    France?")
answer = Answer(["Paris"])
context = [Context(score=0.9, has_answer=True,
    id=1, title="France", text="The capital is
    Paris.")]
document = Document(question, answer, context)
custom_dataset = Dataset(retriever="custom",
    dataset_name="MyDataset")
custom_dataset.documents = [document]
custom_dataset.save_dataset("my_dataset.json")
```

Listing 3: Creating a custom dataset in Rankify.

Loading Structured or QA-Only Datasets Listing 4 presents two alternative dataset loading functions: (1) load_dataset() handles structured formats where each entry includes a query, answer, and retrieved context. (2) load_dataset_qa() is designed for QA-only files (e.g., .jsonl) containing only questions

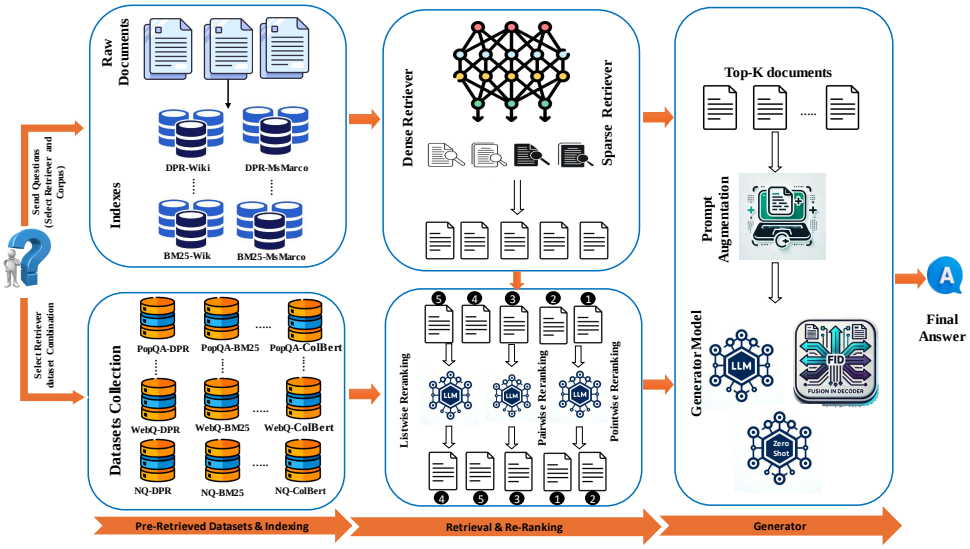

Figure 6: Overview of the RANKIFY pipeline, supporting both real-time and pre-retrieved document retrieval. Users can query documents online via Craw4AI or use pre-retrieved datasets indexed with dense (e.g., DPR) or sparse (e.g., BM25) retrievers, enhanced by auto-indexing with FAISS and API embeddings (e.g., GPT) for corpora like Wikipedia and MS MARCO. The Retrieval & Re-Ranking stage employs dense, sparse, or online methods, followed by pointwise, pairwise, or listwise re-ranking with LLMs. The Generator stage uses advanced RAG methods (e.g., Fusion-in-Decoder) to produce contextually accurate answers.

and answers without retrieval results. This flexibility ensures compatibility with standard QA datasets (e.g., SQuAD) and enables integration with external collections.

```
from rankify.dataset.dataset import Dataset
retrieval_dataset = Dataset.load_dataset("path/to/retrieval_dataset.json")
qa_dataset = Dataset.load_dataset_qa("path/to/qa_dataset.jsonl")
```

Listing 4: Loading a dataset with documents and a QA-only dataset.

Listing 5 demonstrates how to instantiate a retrieval pipeline and execute document retrieval. The example creates a list of Document queries and applies a ColBERT-based retriever. The retriever is initialized with parameters for backend model (colbert-ir/colbertv2.0), number of retrieved documents (n_docs=5), and the index source (e.g., "msmarco"). Retrieved documents are returned with relevance scores and metadata, enabling immediate evaluation or downstream re-ranking.

```
from rankify.dataset.dataset import Document, Question, Answer
from rankify.retrievers.retriever import Retriever

# Define sample documents
documents = [
    Document(question=Question("the cast of a good day to die hard?"),
             answers=Answer(["Jai Courtney", "Bruce Willis"]), contexts=[]),
    Document(question=Question("Who wrote Hamlet?"),
             answers=Answer(["Shakespeare"]), contexts=[])
]
# Initialize a retriever (example: ColBERT)
retriever = Retriever(method="colbert", model="colbert-ir/colbertv2.0", n_docs=5, index_type="msmarco")
# Retrieve documents
retrieved_documents = retriever.retrieve(documents)
# Print retrieved documents
for i, doc in enumerate(retrieved_documents):
    print(f"\nDocument {i+1}:")
    print(doc)
```

Listing 5: Using a retriever model in Rankify to retrieve documents.

Listing 6 showcases how to re-rank documents using transformer-based models. It combines the retrieval logic from Listing 5 with a re-ranker instantiation (Reranker). The example uses the MonoT5-base model to score the retrieved contexts, and returns a reordered list of documents based on relevance. This listing illustrates the modularity of RANKIFY where retrieval and ranking components are decoupled but interoperable.

```
from rankify.dataset.dataset import Document, Question, Answer
from rankify.retrievers.retriever import Retriever
from rankify.rerankers.reranker import Reranker
# Define sample documents
documents = [
    Document(question=Question("the cast of a good day to die hard?"),
             answers=Answer(["Jai Courtney", "Bruce Willis"]), contexts=[]),
    Document(question=Question("Who wrote Hamlet?"),
```

```
            answers=Answer(["Shakespeare"]),
                contexts=[])
]
# Initialize a retriever
retriever = Retriever(method="colbert", model="colbert-ir/colbertv2.0", n_docs=5, index_type="msmarco")
retrieved_documents = retriever.retrieve(documents)
# Initialize a re-ranker
reranker = Reranker(method="monot5",model_name="monot5-base")
# Apply re-ranking
reranked_documents = reranker.rerank(retrieved_documents)
# Print re-ranked documents
for i, doc in enumerate(reranked_documents):
    print(f"\nDocument {i+1}:")
    print(doc)
```

Listing 6: Applying re-ranking in Rankify.

Listing 7 introduces the Generator module, which performs RAG using retrieved contexts. The method demonstrated here uses the "in-context-ralm" generation strategy in combination with a specific LLM backend (meta-llama/Meta-Llama-3.1-8B-Instruct) via Hugging Face. This listing shows how RANKIFY transforms retrieval results into a generation-ready format and allows users to synthesize natural language answers informed by ranked context.

```
from rankify.dataset.dataset import Document, Question, Answer, Context
from rankify.generator.generator import Generator
# Sample question and contexts
question = Question("What is the capital of France?")
answers=Answer('')
contexts = [
    Context(id=1, title="France", text="The capital of France is Paris.", score=0.9),
    Context(id=2, title="Germany", text="Berlin is the capital of Germany.", score=0.5)
]
# Create a Document
doc = Document(question=question, answers=answers, contexts=contexts)
# Initialize Generator
generator = Generator(method="in-context-ralm", model_name='meta-llama/Meta-Llama-3.1-8B-Instruct', backend="huggingface")
# Generate answer
generated_answers = generator.generate([doc])
print(generated_answers)  # Output: ["Paris"]
```

Listing 7: Applying Retrieval-Augmented Generation (RAG) in Rankify.

Listing 8 computes Top-k retrieval metrics—such as Top-1, Top-5, Top-10, etc.—both before and after applying the re-ranking model. This allows users to quantify the marginal gain introduced by rerankers and compare multiple retrieval-reranking configurations. The metric class (Metrics) abstracts over document lists and provides a unified evaluation interface.

```
from rankify.metrics.metrics import Metrics
metrics = Metrics(documents)
# Compute Top-k accuracy before and after re-ranking
before_ranking = metrics.calculate_retrieval_metrics(ks=[1,5,10,20,50,100], use_reordered=False)
after_ranking = metrics.calculate_retrieval_metrics(ks=[1,5,10,20,50,100], use_reordered=True)
print("Before Ranking:", before_ranking)
print("After Ranking:", after_ranking)
```

Listing 8: Computing Top-k accuracy for retrieval and re-ranking in Rankify.

Listing 9 shows how to evaluate answer quality for RAG outputs using generation-based metrics such as Exact Match (EM), precision, recall, and containment. This is particularly relevant for open-domain QA settings, where the generated response must match the gold-standard answer with high fidelity. The calculate_generation_metrics() function receives the generated answers and computes relevant scores.

```
from rankify.metrics.metrics import Metrics
qa_metrics = Metrics(documents)
qa_results = qa_metrics.calculate_generation_metrics(generated_answers)
print(qa_results)
```

Listing 9: Computing QA and RAG evaluation metrics in Rankify.