Large Language Models for Information Retrieval: A Survey

Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, H. Chen, Z. Dou, and J.-R. Wen

---

Manuel Di Agostino

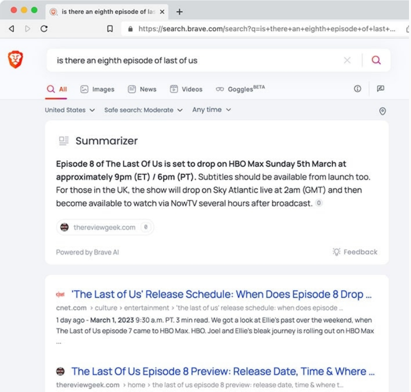
May 29, 2025

University of Parma

Today, do you search on Google or ask ChatGPT?

Introduction

- Traditional IR: retrieve and rank documents.
- LLM-powered IR: understand, summarize, and answer.
- Tools like Bing, Brave, Perplexity, and Gemini integrate LLMs to deliver instant summaries.
- **Users now expect answers, not just links.**



Example: Brave Search summarizing results via LLM.

Why This Survey Matters

- LLMs are transforming not just how we generate language, but how we **access and retrieve information**.
- Classical IR relies on indexing and keyword matching — effective, but limited in understanding intent.
- In contrast, LLMs enable:
    - semantic understanding,
    - multi-turn conversational context,
    - and end-to-end answer generation.

From IR Pipeline to LLM-enhanced Modules

- This survey focuses on how LLMs enhance the four core components of IR:
    1. **Query Rewriter**
    2. **Retriever**
    3. **Reranker**
    4. **Reader**
- Each module faces new opportunities — and new challenges — with the advent of LLMs.

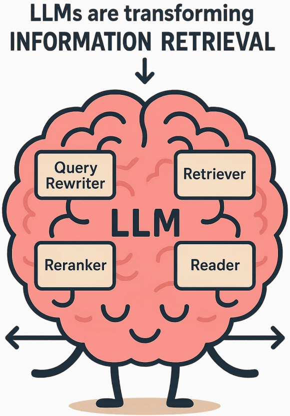


Illustration of modular IR pipeline adapted to LLMs. AI generated.

Background

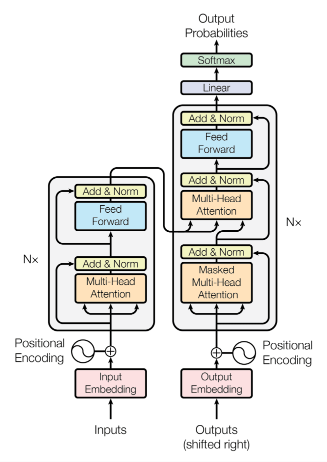
From Classical IR to Neural Retrieval

- Traditional IR systems[1] rely on:
  - keyword matching (e.g., Boolean models [1], BM25 [2]),
  - vector space models (cosine similarity),
  - statistical models (language models).
- Neural IR improves by leveraging:
  - learned dense embeddings,
  - pre-trained language models (e.g., BERT [3]).
- LLMs extend this further: beyond matching, toward **understanding and generation**.

[1] Lectures 3.1-3.4, BDDM A.A. 2024/25, F. **Bertini**

Large Language Models: A Quick Overview

- LLMs are transformer-based models [4] with billions of parameters.
- Key types:
  - **Encoder-only** – understanding
  - **Decoder-only** – generation
  - **Encoder-decoder** – flexible
- Learning styles:
  - **In-context learning**
  - **Fine-tuning**
  - **RAG (retrieval-augmented generation)**



Source: Vaswani et al., "Attention Is All You Need", 2017.

Query Rewriting

- First step in the IR pipeline: improve the quality of the user query.
- Classical techniques: query expansion, pseudo-relevance feedback.
- LLMs allow rewriting queries using:
  - **Prompting**: zero/few-shot style reformulation.
  - **Fine-tuning**: domain-specific transformations.
  - **Knowledge distillation**: compress LLM behavior into smaller models.
- Particularly useful in:
  - *ad-hoc* search with ambiguous queries,
  - *multi-turn* conversational search.

Query2Doc: Few-shot Pseudo-Document Generation

- Query2Doc [5] reframes query rewriting as **text generation**.
- It uses **few-shot prompting** (in-context learning) to guide the LLM.
- Prompt examples are drawn from the MSMARCO dataset [6].
- The model generates a *pseudo-document* that simulates a relevant passage, used to retrieve real documents more effectively.

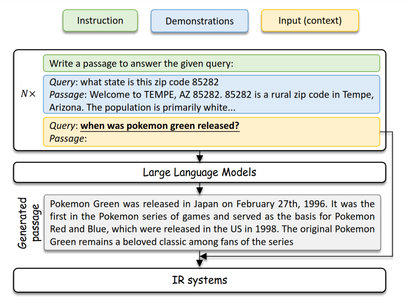


Figure from the original paper.

- **Concept drift** [7, 8, 9]:
  - LLMs may inject unrelated details when rewriting queries.
  - This can dilute the core intent of the original question.
  - Often caused by the LLM's tendency to be verbose or over-informative.
- **Retrieval performance degradation** [10]:
  - Expansion improves weak retrievers, but often **harms stronger ones**.
  - Expansion may help align queries with the expected format when the corpus diverges from the training distribution.
- **Key takeaway:**
  - Query rewriting must be **target-aware** and retriever-aware.
  - More rewriting $\neq$ better results.

Retriever

- The retriever selects candidate documents likely to be relevant to a query.
- **Classical methods**:
  - Sparse retrievers — keyword-based (e.g., BM25 [2]).
  - Dense retrievers — neural representations (e.g., DPR [11]).
- **LLMs improve retrieval** in two complementary ways:
  1. **Data augmentation** — generate synthetic queries and labels for dense retrievers.
  2. **Model enhancement** — build better retrievers using LLM architectures.

LLMs for Data Augmentation in Retrieval

- **Motivation:** manual annotation of training data is expensive and domain-specific.
- LLMs can generate synthetic training signals:
  - **Pseudo-query generation:** generate questions for existing documents (e.g., InPairs [12] + GPT-3 [13]).
  - **Relevance label generation:** assign soft relevance scores to query-document pairs (e.g., ART [14]), used as training targets for dense retrievers.
- Enables few-shot and zero-shot retrieval training across domains.

- LLMs can serve as the **retriever itself**, not just as a data generator.
- Three main approaches:
  1. **Dense retrievers:** use LLMs as encoders to map queries and documents into vector space (e.g., GTR [15], RepLLaMA [16])
  2. **Task-aware retrievers:** prepend task-specific instructions to queries to guide retrieval (e.g., TART [17])
  3. **Generative retrievers:** LLM decodes document identifiers directly from queries (e.g., DSI [18], LLM-URL [19])
- These models leverage LLMs' semantic understanding for more accurate and flexible retrieval.

Reranker

Reranker: Second-pass Filtering

- The reranker receives the candidate documents from the retriever.
- It refines the ranking by evaluating the **query-document relevance** more precisely.
- With LLMs, three usage paradigms emerge:
    1. Supervised rerankers
    2. Unsupervised rerankers
    3. LLM-assisted data augmentation
- **Goal:** assign better scores → improve top-ranked results.

- LLMs are fine-tuned on labeled datasets (e.g., MSMARCO) to learn relevance signals.
- Three architectural types:
  1. **Encoder-only:** monoBERT [20] uses the embedding for scoring (`[CLS] query [SEP] document [SEP]`).
  2. **Encoder-decoder:** T5 [21] generates a classification token (`true`/`false`).
  3. **Decoder-only:** RankLLaMA [22] formats input as a prompt (`query: {query} document: {document} [EOS]`) and uses the last token's embedding.
- Loss functions: cross-entropy, pairwise, listwise.

Unsupervised Reranking with LLMs

- Large LLMs (10B+ params) make fine-tuning difficult, so prompting is used for unsupervised reranking.
- Three main methods:
  1. **Pointwise**: Score each query-document pair independently. **Open-source models required**: to access the logits of the ”YES” and ”NO” tokens.
  2. **Listwise**: rank a list of documents at once; better accuracy but costly and sensitive to input order.
  3. **Pairwise**: compare document pairs to build ranking; good accuracy but computationally expensive.
- Prompt engineering and few-shot examples improve results.

Reader

- The reader generates answers from top-ranked documents retrieved by the IR system.
- Reader models differ in how they interact with the retrieval process:
  1. **Passive Readers** — receive documents from the IR system and generate answers.
     - ⋆ *Once-Retrieval* (e.g., RAG [23]): retrieve once at the beginning.
     - ⋆ *Periodic-Retrieval* (e.g., RETRO [24]): retrieve during generation (every n tokens).
     - ⋆ *Aperiodic-Retrieval* (e.g., FLARE [25]): retrieve when confidence is low.

LLM-Based Reader: Typologies and Strategies II

2. **Active Readers** — LLMs autonomously decide when and what to retrieve.
   - Formulate follow-up queries (e.g., Self-Ask [26]);
   - Build reasoning chains across retrieval iterations;
3. **Compressors** — reduce retrieved content to fit LLM input limits.
   - *Extractive* (e.g., LeanContext [27]) or *abstractive* (e.g., TCRA [28]) compression.

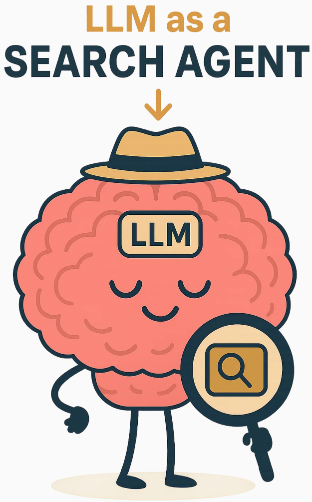
- These strategies balance accuracy, interactivity, and computational efficiency.

Retrieval-Augmented Generation (RAG)

- **RAG** [23] integrates a retriever and a generator in a single architecture.
- At inference time:
  - A retriever selects top-k documents given a query.
  - A generator (LLM) conditions on both query and documents to produce an answer.
- Advantages:
  - Combines factual grounding (retriever) with fluent generation (LLM).
  - Allows open-book reasoning with up-to-date information.
- Limitation: risk of hallucinating content not grounded in the retrieved passages.

Search Agents

- **Goal:** mimic human browsing to search, interpret, and synthesize autonomously.
- **WebGPT** [29]
  - Answers questions via web browsing
  - Cites sources; reward model encourages factuality
- **ReAct** [30]
  - Interleaves *Thought* and *Action*
  - Generates reasoning steps and search commands



LLM as a search agent. AI generated.

Conclusion

Looking Ahead: Future Directions

- **Query rewriting**: improve personalization and reward-aware reformulation.
- **Retriever**: reduce latency, support multimodal and updatable indexes.
- **Reranker**: enhance online efficiency and adapt to diverse ranking tasks.
- **Reader**: increase factuality and snippet selection to avoid hallucinations.
- **Evaluation**: go beyond relevance—measure generation quality and faithfulness.

References

[1] C. J. van Rijsbergen. *Information Retrieval.* 2nd. London, UK: Butterworths, 1979.

[2] S. E. Robertson et al. "Okapi at TREC-3". In: *Proceedings of The Third Text REtrieval Conference (TREC-3)*. Ed. by D. K. Harman. Vol. 500-225. NIST Special Publication. Gaithersburg, Maryland, USA: National Institute of Standards and Technology (NIST), 1994, pp. 109–126.

[3] Jacob Devlin et al. **“BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”.** In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Volume 1 (Long and Short Papers)*. Ed. by Jesse Burstein, Chris Doran, and Thamar Solorio. Minneapolis, MN, USA: Association for Computational Linguistics, June 2019, pp. 4171–4186.

[4] Ashish Vaswani et al. ***Attention Is All You Need.*** 2023. arXiv: `1706.03762 [cs.CL]`. URL: `https://arxiv.org/abs/1706.03762`.

[5] Liang Wang, Nan Yang, and Furu Wei. ***Query2doc: Query Expansion with Large Language Models.*** 2023. arXiv: `2303.07678 [cs.IR]`. URL: `https://arxiv.org/abs/2303.07678`.

[6] T. Nguyen et al. **"MS MARCO: A Human Generated Machine Reading Comprehension Dataset".** In: *Proceedings of the Workshop on Cognitive Computing (CoCo@NIPS)*. Vol. 1773. CEUR Workshop Proceedings. CEUR-WS.org, 2016. URL: `http://ceur-ws.org/Vol-1773/CoCoWS2016_paper16.pdf`.

[7] A. Anand et al. **"Context-aware Query Rewriting for Text Rankers using LLM".** In: *CoRR* abs/2308.16753 (2023). URL: `https://arxiv.org/abs/2308.16753`.

[8] K. D. Dhole, R. Chandradevan, and E. Agichtein. **"An Interactive Query Generation Assistant using LLM-based Prompt Modification and User Feedback".** In: *CoRR* abs/2311.11226 (2023). URL: `https://arxiv.org/abs/2311.11226`.

[9] W. Peng et al. **"Large Language Model Based Long-Tail Query Rewriting in Taobao Search".** In: *CoRR* abs/2311.03758 (2023). URL: `https://arxiv.org/abs/2311.03758`.

[10] O. Weller et al. **“When Do Generative Query and Document Expansions Fail? A Comprehensive Study Across Methods, Retrievers, and Datasets”.** In: *CoRR* abs/2309.08541 (2023). URL: https://arxiv.org/abs/2309.08541.

[11] Vladimir Karpukhin et al. **“Dense Passage Retrieval for Open-Domain Question Answering”.** In: *CoRR* abs/2004.04906 (2020). URL: https://arxiv.org/abs/2004.04906.

[12] L. H. Bonifacio et al. **“InPars: Data Augmentation for Information Retrieval Using Large Language Models”.** In: *CoRR* abs/2202.05144 (2022). URL: https://arxiv.org/abs/2202.05144.

[13] Tom B. Brown et al. **"Language Models are Few-Shot Learners".** In: *Advances in Neural Information Processing Systems 33 (NeurIPS)*. Ed. by Hugo Larochelle et al. 2020, pp. 1877–1901. URL: `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[14] D. S. Sachan et al. **"Questions Are All You Need to Train a Dense Passage Retriever".** In: *Transactions of the Association for Computational Linguistics (TACL)* 11 (2023), pp. 600–616. URL: `https://aclanthology.org/2023.tacl-1.30`.

[15] Jingfei Ni et al. **"Large Dual Encoders are Generalizable Retrievers".** In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2022, pp. 9844–9855. URL: `https://aclanthology.org/2022.emnlp-main.681`.

[16] Xia Ma et al. **"Fine-tuning LLaMA for Multi-Stage Text Retrieval".** In: *CoRR* abs/2310.08319 (2023). URL: https://arxiv.org/abs/2310.08319.

[17] Akari Asai et al. **"Task-Aware Retrieval with Instructions".** In: *Findings of the Association for Computational Linguistics: ACL 2023*. Ed. by Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, 2023, pp. 3650–3675. URL: https://aclanthology.org/2023.findings-acl.229.

[18] Yao Tay et al. **"Transformer Memory as a Differentiable Search Index".** In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022. URL: https://proceedings.neurips.cc/paper/2022/hash/892840a6123b5ec99ebaab8be1530fba-Abstract-Conference.html.

[19] Nathan Ziems et al. **"Large language models are built-in autoregressive search engines".** In: *Findings of the Association for Computational Linguistics: ACL 2023*. Ed. by Amy Rogers, Jacob Boyd-Graber, and Naoki Okazaki. Toronto, Canada, July 2023, pp. 2666–2678.

[20] Rodrigo F. Nogueira et al. **"Multi-stage document ranking with BERT".** In: *CoRR* abs/1910.14424 (2019). arXiv: `1910.14424`. URL: `https://arxiv.org/abs/1910.14424`.

[21] Colin Raffel et al. **"Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer".** In: *Journal of Machine Learning Research* 21 (2020), 140:1–140:67.

[22] Xinyu Ma et al. **"Fine-tuning LLaMA for Multi-Stage Text Retrieval".** In: *CoRR* abs/2310.08319 (2023). arXiv: `2310.08319`. URL: `https://arxiv.org/abs/2310.08319`.

[23] Patrick S. H. Lewis et al. **“Retrieval-augmented generation for knowledge-intensive NLP tasks”.** In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Virtual*. Ed. by Hugo Larochelle et al. Dec. 2020.

[24] Sebastian Borgeaud et al. **“Improving Language Models by Retrieving from Trillions of Tokens”.** In: *Proceedings of the 39th International Conference on Machine Learning (ICML 2022), Baltimore, Maryland, USA, July 17–23, 2022*. Ed. by Kannan Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 2206–2240.

[25] Zhengbao Jiang et al. **"Active Retrieval Augmented Generation".** In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 7969–7992. DOI: `10.18653/v1/2023.emnlp-main.495`. URL: `https://aclanthology.org/2023.emnlp-main.495/`.

[26] Ofir Press et al. **"Measuring and Narrowing the Compositionality Gap in Language Models".** In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 5687–5711.

[27] Haozhe Liu et al. **"LeanContext: Token-Efficient Context Compression for Retrieval-Augmented Generation".** In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 2023.

[28] Wenkai Yu et al. **"TCRA: Tree-based Context Reduction for Abstractive Question Answering".** In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. 2023.

[29] Reiichiro Nakano et al. **"WebGPT: Browser-assisted question-answering with human feedback".** In: *OpenAI* (2021). `https://openai.com/research/webgpt`.

[30] Shinn Yao et al. **"ReAct: Synergizing reasoning and acting in language models".** In: *arXiv preprint arXiv:2210.03629* (2022).

[31] J. Martineau and T. Finin. **"Delta TFIDF: An Improved Feature Space for Sentiment Analysis".** In: *Proceedings of the Third International Conference on Weblogs and Social Media (ICWSM 2009)*. Ed. by E. Adar et al. San Jose, California, USA: The AAAI Press, May 2009.