Optimizing Retrieval-Augmented Generation for E-Commerce How-To Assistance

Gilad Fuchs and Leonid Ekimov
eBay Inc.
Amsterdam, The Netherlands
{gfuchs, lekimov}@ebay.com

Fei Dong
eBay Inc.
Berlin, Germany
fedong@ebay.com

Jia Hong Xie
eBay Inc.
Shanghai, China
jihoxie@ebay.com

David Liu
eBay Inc.
Zurich, Switzerland
wliu7@ebay.com

Maxim Manco and Alex Nus
eBay Inc.
New York, United States
{mmanco, alnus}@ebay.com

Abstract

Conversational AI is increasingly used at eBay to deliver personalized customer support. We present a production RAG-based How-To Assistant that answers support and how-to queries by grounding responses in a proprietary knowledge base. We study three factors that drive quality: (1) document chunking and contextualization for indexing, (2) query refinement methods, and (3) automatic LLM-based evaluation for rapid iteration and reliable measurement. We also describe the end-to-end system workflow - from offline indexing to real-time serving and report deployment metrics, offering practical guidance for building scalable, high-precision RAG assistants in commercial support settings.

1 Introduction

Conversational AI is increasingly adopted in e-commerce to provide personalized, real-time customer support. Recent advances in large language models (LLMs), particularly instruction-tuned models, have significantly improved Retrieval-Augmented Generation (RAG), enabling assistants that can reliably ground responses in external knowledge sources while maintaining high-quality natural language interaction (Ouyang et al., 2022; OpenAI, 2023). Earlier work demonstrated the effectiveness of combining retrieval with generation for knowledge-intensive tasks (Lewis et al., 2020).

We study a *How-To Assistant*, a specialized conversational sub-agent that answers customer support and "how-to" questions using RAG over a proprietary knowledge base of customer service and seller guides. Despite recent progress, deploying RAG in this setting remains non-trivial. In practice, system performance is highly sensitive to design choices in document chunking, retrieval, and query formulation, and robust evaluation remains challenging due to coupled retrieval–generation failure modes (Wang et al., 2024).

In this work, we present an empirical study of three factors that most affect production performance: (1) document chunking and contextualization strategies for indexing support content, (2) query refinement methods that improve retrieval quality during multi-turn interactions, and (3) an automatic LLM-based evaluation framework that enables rapid iteration and reliable measurement at scale (Es et al., 2023). We further describe the end-to-end production pipeline and report real-world performance metrics from deployment, offering practical guidance for building scalable, high-precision RAG assistants in commercial support domains.

2 Related Work

E-commerce platforms have long used conversational agents for customer support, evolving from rule-based systems to neural retrieval and generation. AliMe Chat (Qiu et al., 2017) combined retrieval with generation at scale, while task-oriented shopping assistants highlighted challenges such as domain coverage and contextual reasoning (Yan et al., 2017). Product QA work further emphasizes heterogeneous evidence and subjective user needs (Carmel et al., 2018; Deng et al., 2023).

RAG-based assistants depend on accurate retrieval: lexical methods are strong baselines (Robertson and Zaragoza, 2009), while dense re-

trieval improves semantic matching for natural language queries (Karpukhin et al., 2020). Hybrid and late-interaction approaches further improve precision via reranking (Khattab and Zaharia, 2020). RAG then conditions generation on retrieved evidence to improve factuality (Lewis et al., 2020), with variants such as REALM (Guu et al., 2020) and FiD (Izacard and Grave, 2021) exploring tighter retrieval–generation coupling; grounding also helps reduce hallucinations in support dialogue (Dinan et al., 2019; Islam et al., 2024).

Despite this progress, prior work often reports limited end-to-end evidence on how indexing choices, query rewriting, and evaluation protocols interact in deployed assistants. Our work complements the literature by providing a large-scale, deployment-grounded analysis of these design trade-offs in a commercial support setting.

3 RAG System Design

We describe the online inference flow and the offline indexing pipeline.

Online Flow Given a user message routed to support, a Dialogue Manager LLM predicts (i) whether retrieval is needed (*searchable*), (ii) whether the user requests a live agent, and (iii) a rewritten retrieval query. If *searchable*, we embed the rewritten query and retrieve the top-K chunks via vector search. A Responder LLM then generates the final answer conditioned on the user message and retrieved evidence, returning citations/links.

The Responder LLM also classifies the outcome as *clear*, *ambiguous*, or *no-answer*: for ambiguous cases (e.g., buyer question vs. seller question), it defaults to the more common Buyer case and appends a role-clarifying follow-up question; for no-answer cases it triggers a fallback such as clarification or agent handoff (Figure 1).

Offline Documents Processing The system indexes buyer and seller facing support documentation used to answer how-to queries. A daily offline pipeline ingests the content, segments documents into chunks, computes embeddings, and stores text and vectors for retrieval at inference time. Details of the chunking strategies and embedding models are described in subsequent sections.

4 Experimental Setup

We evaluate components separately when possible and validate changes end-to-end on synthetic and real interaction logs.

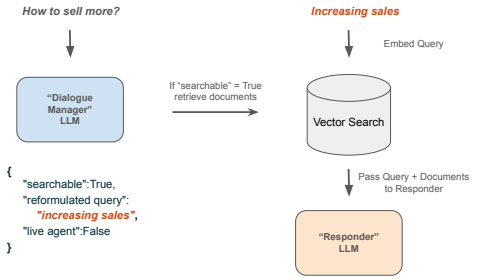

Figure 1: Online flow: classify, rewrite, retrieve, and generate; ambiguous/no-answer cases trigger clarification or fallback.

4.1 Datasets

In this section, we introduce the documents and datasets utilized for both the building retrieval index and the evaluation process.

4.1.1 Documents Corpus

The retrieval corpus comprises two collections: customer support help articles and seller-facing documentation. Together, they cover a broad range of topics, including buying and selling workflows, account management, payments, and troubleshooting, as well as seller-specific guidance such as account setup, marketing, and seller protections. Many guides exhibit heterogeneous structures, often spanning multiple topics and including embedded Q&A sections.

Statistic	Customer Service Articles	Seller Center
Number of Documents	449	146
Avg Doc. Length (words)	760	750
Avg Doc. Length (tokens, est.)	∼1000	∼975
Max Doc. Length (words)	17101	4066
Max Doc. Length (tokens, est.)	∼22200	∼5300
Min Doc. Length (words)	40	33
Min Doc. Length (tokens, est.)	∼50	∼40

Table 1: Documents Index Statistics (token counts are approximate, assuming ∼1.3 tokens per word)

4.1.2 Evaluation Dataset

We evaluate retrieval and generation using two datasets:

Synthetic Dataset We generate up to five question–answer pairs per document using an LLM, yielding 2,645 pairs with ground-truth source URLs. This dataset is used to evaluate retrieval quality, particularly for comparing chunking strategies.

To ensure quality, we enforce that all generated question–answer pairs are strictly grounded in the source document, with the originating document URL used as retrieval ground truth. We further increase dataset diversity by generating paraphrased variants of each question to simulate real user query variability. Finally, we manually inspect a sample of the generated pairs to verify correctness and alignment with the source content.

Real Assistant Logs We use real assistant conversations covering buyer and seller support scenarios. These logs are used for end-to-end and production-oriented evaluation. A subset of 300 interactions is manually annotated by members of the research team, with one annotation per interaction, indicating whether a query is *searchable* and whether it requires escalation to a *live agent*. These annotations are used to evaluate system routing behavior and to support error analysis. In addition, trained annotators following structured guidelines are used in a separate evaluation setting to assess response quality and alignment with retrieved context. These annotations are distinct from the routing annotations described above and are used for metric validation (Section 4.5). Table 2 shows representative examples illustrating the ambiguity and overlap between the two labels.

User message	Searchable	Live agent
I would like to speak to a live agent	False	True
How do I get started selling on the platform live?	True	False
Hi, I have a problem	False	False
Returned an item 11 days ago and no refund; I can't speak to anyone	True	True

Table 2: Illustrative examples of routing labels in the Real Assistant Logs dataset.

4.2 Embedding Models

We evaluate two embedding models: MPNet-v2[1], an open-source embedding model with a 384-token context window, and Vector-Prime, an internally developed 1B parameters embedding model supporting up to 2048 tokens. Vector-Prime is fine-tuned on internal e-commerce data and optimized for long-context document and query encoding, enabling richer representations for support-oriented retrieval.

[1]https://huggingface.co/sentence-transformers/all-mpnet-base-v2

4.3 Chunking Strategies

Chunking is required to accommodate embedding context limits and preserve relevant information for retrieval. We compare fixed-size, sentence-based, semantic, and recursive chunking strategies. Recursive chunking, following the approach implemented in LangChain (Chase, 2022) iteratively segments documents using structural cues to maintain semantic coherence while allowing flexible chunk sizes. In addition, we evaluate metadata enrichment by appending titles or keywords to each chunk. This enhances the retrieval context by providing supplementary information, though it introduces added complexity to the chunking process and requires careful management of the metadata.

Results show that recursive chunking achieves the strongest retrieval performance, with additional gains from metadata enrichment (Section 5.1.2).

4.4 Query Refinement

To improve retrieval quality, we evaluate several query reformulation strategies applied by the Dialogue Manager. These include: (i) query simplification, which removes extraneous phrasing to focus on the core intent; (ii) adding explicit details to enrich underspecified queries, where the Dialogue Manager LLM infers missing context (e.g., user role, task stage, or relevant entities) and expands the query into a more fully specified form aligned with support document structure, e.g., a short query such as *"cancel order"* may be expanded into *"how to cancel an order on eBay, including eligibility, timing, and refund conditions"*; and (iii) keyword expansion, which augments the query with salient terms relevant to the user's intent. We also explore generative reformulations based on Hypothetical Document Embeddings (HyDE) (Gao et al., 2023), where a simulated answer is appended to the query to better match the language and structure of support documents, e.g., a query such as *"track my buying activity"* may be expanded with a simulated answer describing access to purchase history, order details, and shipment tracking. In the most expressive variant, we further combine an explicit representation of user intent with the simulated answer, yielding intent-aware, answer-shaped queries. In practice, these rewrites introduce additional semantically related terms and make implicit aspects of the query explicit, improving alignment with document language. While such transformations can occasionally alter phrasing or introduce minor

assumptions, we observe that the original intent is generally preserved, with gains in retrieval relevance outweighing potential semantic drift. These reformulation strategies trade off brevity and contextual richness, aiming to improve semantic alignment between user queries and indexed document chunks. Section 5.1.3 reports a quantitative comparison of their impact on retrieval performance.

4.5 Automatic Evaluation and Metrics

Manual evaluation does not scale for large conversational systems, so we adopt an automated *LLM-as-a-Judge* framework (Gu et al., 2025) to assess retrieval quality, response quality, and end-to-end behavior. The judge is a GPT-4o model run with temperature 0, invoked once per example (single-pass) to ensure determinism. It scores each response using a structured JSON output.

Metrics For retrieval, we report LLM-judged Relevance scores on a 1-10 scale, along with Hit@K and Mean Average Precision (mAP). For response quality, we use the judge's Answer Relevancy score on a 1-5 scale, reflecting how well the generated answer addresses the user query. We consider a response to have *Helped User* if its Answer Relevancy score is ≥ 4, and to be a *Low-Quality RAG Answer* otherwise. The threshold was chosen to align with human judgments of satisfactory answers in preliminary calibration.

End-to-End Evaluation We evaluate system-level behavior using an end-to-end (E2E) confusion matrix that distinguishes correct assistance (*Helped User*, defined as Answer Relevancy ≥ 4), correct rejection of non-searchable queries, and error cases including low-quality answers, retrieval failures, and searchability misclassification (Figure 2). Based on this matrix, we define E2E Accuracy as:

$$\text{E2E Accuracy} = \frac{\text{Helped} + \text{Correctly Rejected}}{\text{Total Queries}}$$

5 Results

In this section, we provide a detailed evaluation of the How-To Assistant, examining its performance across various components and configurations. We assess retrieval quality, evaluate the impact of query refinement techniques, and compare human and automatic evaluations to validate the system's accuracy and reliability. Our focus is on controlled analysis of individual RAG components within a consistent system, rather than comparing end-to-end pipelines with differing architectures, allowing us to isolate the impact of key design choices under production constraints. Finally, we discuss the system's deployment in production and its overall impact on user query resolution.

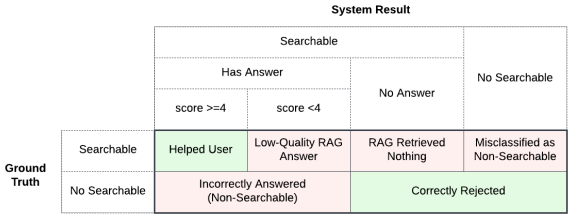

Figure 2: E2E confusion matrix comparing ground-truth labels and system result

Method	Hit@1	Hit@2	Hit@3	Hit@4	Hit@5	mAP
Vector-Prime	0.527	0.657	0.726	0.765	0.791	0.630
MPNet-v2	0.516	0.653	0.717	0.760	0.788	0.622

Table 3: Retrieval performance of embedding models on the synthetic dataset under fixed-size chunking (384 tokens, 50-token overlap).

5.1 Retrieval

We evaluate retrieval performance on the synthetic dataset (Section 4.1.2), which contains 2,645 generated question-answer pairs. For each query, the dataset provides one or more ground-truth source URLs. A retrieved chunk is counted as correct if its originating URL matches any ground-truth URL for that query.

We report Hit@K (whether at least one of the top-K retrieved chunks is correct) and mean Average Precision (mAP) over the ranked retrieval list. We conduct ablations over (i) embedding model, (ii) chunking strategy, (iii) metadata enrichment, and (iv) chunk size. Unless otherwise stated, we use an overlap of 50 tokens between adjacent chunks.

5.1.1 Embedding Models

To isolate the effect of the embedding model, we compare MPNet-v2 and Vector-Prime under a fixed chunking configuration (384 tokens with a 50-token overlap). As shown in Table 3, Vector-Prime yields small but consistent improvements over MPNet-v2 across all Hit@K metrics and mAP.

5.1.2 Chunking

Next, we examine how chunking strategy affects retrieval quality. To keep the embedding model constant while comparing chunkers, we run this ablation with MPNet-v2 and a target chunk length of 384 tokens (50-token overlap where applicable). Table 4 compares fixed-size, sentence-based, semantic, and recursive chunking. Recursive chunking achieves the strongest overall performance (highest Hit@1 and mAP), suggesting that preserving document structure while allowing flexible boundaries can improve retrieval effectiveness.

Chunker	Hit@1	Hit@2	Hit@3	Hit@4	Hit@5	mAP
Fixed-Size	0.516	0.653	0.717	0.76	0.788	0.622
Sentence	0.514	0.647	0.712	0.749	0.777	0.617
Semantic	0.526	0.663	0.729	0.77	0.793	0.632
Recursive	0.535	0.664	0.726	0.77	0.796	0.636

Table 4: Retrieval performance of chunking strategies using MPNet-v2 with a target chunk length of 384 tokens and 50-token overlap (where applicable).

Metadata Enrichment We next evaluate whether enriching chunks with lightweight document metadata improves retrieval. Specifically, we concatenate document titles, descriptions, and keywords with each chunk prior to embedding. This metadata is sourced from the webpage (e.g., title/description) or internal fields (e.g., keywords). As shown in Table 5, metadata enrichment substantially improves MPNet-v2 (+0.034 mAP), and yields a smaller but positive mAP gain for Vector-Prime (+0.009 mAP), even though some Hit@K values change only marginally. Overall, titles and descriptions likely help by capturing document themes that better align with query intent and by reducing ambiguity about document scope.

Model	Meta included	Hit@1	Hit@2	Hit@3	Hit@4	Hit@5	mAP
MPNet-v2	N	0.535	0.664	0.726	0.77	0.796	0.636
	Y	0.567	0.702	0.764	0.80	0.829	0.670
Vector-Prime	N	0.531	0.673	0.733	0.77	0.797	0.637
	Y	0.551	0.674	0.736	0.77	0.791	0.646

Table 5: Effect of metadata enrichment under recursive chunking (target length 384 tokens).

Chunk Size Last, we study the effect of chunk size under recursive chunking using Vector-Prime, which supports longer inputs than MPNet-v2. Keeping the overlap fixed at 50 tokens and including metadata, Table 6 shows that increasing the target chunk length beyond 384 tokens degrades performance in this setting. Thus, among the sizes tested, a target chunk length of 384 tokens performs best for Vector-Prime with recursive chunking. Unless otherwise stated, we use Vector-Prime with recursive chunking (384 tokens, 50-token overlap) and metadata for subsequent experiments.

Chunk size	Hit@1	Hit@2	Hit@3	Hit@4	Hit@5	mAP
384	0.551	0.674	0.736	0.77	0.791	0.646
512	0.539	0.666	0.729	0.762	0.781	0.636
1024	0.525	0.651	0.704	0.736	0.765	0.619

Table 6: Effect of target chunk length under recursive chunking with Vector-Prime (50-token overlap, metadata included).

5.1.3 Query Refinement

We next evaluate query refinement on the Real Assistant Logs (Section 4.1.2). Unlike the synthetic dataset, these logs do not provide URL-level ground truth, so we assess retrieval using an LLM-based relevance judge. We use GPT-4o as a Dialogue Manager to rewrite each user query into several refinement variants. Retrieval uses the selected configuration from the ablations above: Vector-Prime embeddings with recursive chunking (384 tokens, 50-token overlap) and metadata enrichment. For each retrieved chunk, an LLM assigns a relevance score from 1-10 with respect to the user query; chunks with scores ≥ 8 are treated as relevant. We then compute Hit@K and mAP based on this binarized relevance over the ranked list. Table 7 shows that combining the original query with both an inferred user intent and a simulated response yields the strongest retrieval performance.

5.2 Human vs. Automatic Evaluation

To validate our automated evaluation framework, we compare human annotations with LLM-based automatic assessments on the same set of assistant outputs. This analysis quantifies how closely the Auto Evaluator tracks human judgment when scoring generated responses and end-to-end system outcomes.

We run this study on a randomly sampled set of 500 Real Assistant Logs (Section 4.1.2). Each query is processed through the full How-To Assistant pipeline, including Vector-Prime for retrieval embeddings, a GPT-4o-mini "Dialogue Manager", and a "RAG Responder". For retrieval, we use recursive chunking with metadata enrichment and a query simplification strategy. To ensure direct

Refinement method	Hit@1	Hit@2	Hit@3	Hit@4	Hit@5	mAP
Simplifying	0.381	0.519	0.565	0.598	0.611	0.476
Refine with Details	0.406	0.570	0.627	0.668	0.676	0.519
Query + Keywords	0.384	0.531	0.567	0.588	0.596	0.476
Query + Simulated Response	0.506	0.635	0.663	0.691	0.711	0.591
Query + Intent + Simulated Response	**0.571**	**0.684**	**0.704**	**0.721**	**0.733**	**0.641**

Table 7: Comparison of query refinement methods on Real Assistant Logs using LLM-judged relevance (relevant if score ≥ 8).

comparability, we evaluate the exact same query-response pairs with both human and automatic scoring.

Three human labelers annotate the metrics defined in Section 4.5. Inter-annotator agreement is high: the average κ across the three labelers exceeds 0.87 for all metrics.

To assess end-to-end behavior, we measure (i) whether the assistant correctly routes queries as *searchable* vs. *non-searchable*, and (ii) whether it produces a satisfactory outcome when an answer is required. Because routing decisions are made exclusively by the Dialogue Manager, the E2E confusion matrix provides a direct evaluation of its classification behavior in conjunction with downstream retrieval and generation. We report the distribution over outcome categories and compute E2E Accuracy as described in Section 4.5. The combined human and automatic results are shown in Table 8.

Metric	Human Eval.	Auto Eval.
Helped User	71.4%	73.8%
Low-Quality RAG Answer	2.4%	1.4%
RAG Retrieved Nothing	9.8%	8.0%
Misclassified as Non-Searchable	0.2%	0.6%
Incorrectly Answered (Non-Searchable)	0.2%	0.8%
Correctly Rejected	16.0%	15.4%
E2E Accuracy	**87.4%**	**89.2%**

Table 8: Human vs. automatic end-to-end evaluation on 500 Real Assistant Logs.

Overall, E2E Accuracy differs by 1.8 percentage points between human and automatic evaluation (87.4% vs. 89.2%). The category-level outcome distribution is also similar (e.g., *Helped User*, *Correctly Rejected*, and the error modes related to retrieval and routing), suggesting that the Auto Evaluator is a reliable proxy for human judgments at the level needed for iterative system development. The automatic evaluator tends to produce slightly higher and more consistent scores due to deterministic prompting and uniform application of evaluation criteria. In contrast, human annotators may exhibit minor variability in borderline cases, leading to small differences in aggregate metrics. In practice, this enables rapid evaluation of both component failures (e.g., retrieval returning nothing) and end-to-end decisions at a scale that would be costly to label manually. We have also used the Auto Evaluator to compare additional system variants beyond those reported in this paper.

5.3 Final Evaluation and Production Performance

After selecting the best-performing retrieval and prompting configuration from the component-level experiments, we run a final human-only quality assessment prior to deployment. For this final setting, we use Vector-Prime embeddings with recursive chunking and metadata enrichment. The "Dialogue Manager" uses GPT-4o-mini with few-shot prompting. Unlike the configuration used in Table 8 (query simplification), the final system applies the query refinement strategy *User query + User Intent + Simulated response*. Table 9 reports the resulting end-to-end outcomes under human evaluation. E2E Accuracy increases from 87.4% (Table 8) to 94.8%, consistent with the gains observed from richer query refinement in our retrieval experiments.

Metric	Human Eval.
Helped User	81%
Low-Quality RAG Answer	1.2%
RAG Retrieved Nothing	1.8%
Misclassified as Non-Searchable	0.4%
Incorrectly Answered (Non-Searchable)	1.6%
Correctly Rejected	13.8%
E2E Accuracy	**94.8%**

Table 9: Final end-to-end evaluation of the How-To Assistant (human annotation).

Given the improved end-to-end performance observed offline, we proceeded to deploy the How-To Assistant to production within the eBay mobile app. In the production setting, How-To responses exhibited strong user engagement when triggered. Specifically, 60.5% of How-To sessions continued with an additional assistant interaction, and 52.6% showed direct engagement (e.g., interacting with assistant UI elements or issuing a follow-up query). Relative to other assistant response types observed during the same evaluation period (Dec 2025 - Jan 2026), How-To responses showed the strongest follow-on engagement. For comparison, general product informational responses led to subsequent assistant interactions in only 44.3% of sessions and

direct engagement in 25.9%, compared to 60.5% and 52.6% for How-To responses, respectively.

To assess whether response quality in production aligns with offline evaluation, we analyzed a sample of production sessions that triggered a How-To response. In production, the How-To Assistant produced a non-default response in approximately 90% of triggered cases, reflecting a high rate of successful answer generation under real-world conditions. Among these responses, 93% were labeled as *Helped User* by the LLM judge, indicating that most surfaced answers were judged to be relevant and useful. For offline comparison, we condition on queries for which the system attempted to respond. Under this comparable slice, 98% of offline responses were labeled as *Helped User* based on human annotation. While offline evaluation benefits from highly controlled conditions, the production results remain close in magnitude, demonstrating robust transfer of response quality from offline evaluation to real-world deployment.

Overall, production analytics indicate that the How-To Assistant combines high response quality with strong engagement when triggered, supporting the hypothesis that How-To Assistance addresses unmet, high-intent informational needs in shopping journeys.

Ethical Considerations Our system operates on proprietary eBay support content and anonymized user interaction logs. We ensure that no personally identifiable information (PII) is used in training or evaluation. The assistant is designed to provide guidance only within supported domains and to defer to human agents when appropriate, reducing the risk of misleading or unsafe responses.

6 Conclusions

In this paper, we investigated the design and optimization of a How-To Assistant. Through extensive experimentation, we identified key design choices that improve RAG performance, including recursive chunking strategies and query refinement techniques.

We further evaluated the system in a real-world production setting and observed strong engagement and high-quality responses when the How-To Assistant was triggered. These findings demonstrate that the proposed design choices transfer effectively from offline evaluation to deployment, providing practical evidence of their robustness under real usage conditions.

Overall, our findings suggest that in support-oriented RAG systems, retrieval quality is strongly driven by indexing and query formulation decisions, often delivering larger gains than changes to core models while preserving serving efficiency. Moreover, calibrated LLM-based automatic evaluation enables rapid iteration during offline development and provides signals that are predictive of production behavior. Together, these results offer actionable guidance for building scalable, high-quality customer support assistants in large e-commerce environments.

We note that more agentic approaches may further improve support assistants, for example through iterative retrieval, tool use, and multi-step decision making. However, such approaches can also introduce additional latency and system complexity, which are important considerations in production settings. This paper therefore focuses on establishing strong baselines and practical design guidance for a production RAG pipeline, leaving a systematic study of agentic methods to future work.

References

David Carmel and 1 others. 2018. Product question answering using customer generated content – research challenges. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery.

Harrison Chase. 2022. Langchain. *GitHub repository*.

Chuan Deng and 1 others. 2023. Product question answering in e-commerce: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. OpenReview.

Shahul Es and 1 others. 2023. Ragas: Automated evaluation of retrieval-augmented generation. *arXiv preprint arXiv:2309.15217*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. A survey on llm-as-a-judge. *Preprint*, arXiv:2411.15594.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. PMLR.

Md Rashadul Islam and 1 others. 2024. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.00123*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open-domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, and 1 others. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Minghui Qiu, Feng Li, Shujie Wang, Jianfeng Gao, Yu Chen, Yang Song, Dengyong Zhao, and Haifeng Chen. 2017. Alime chat: A sequence to sequence and rerank based chatbot engine. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*.

Liang Wang and 1 others. 2024. Searching for best practices in retrieval-augmented generation. *Proceedings of the EMNLP*.

Rui Yan, Yang Song, and Hua Wu. 2017. Building task-oriented dialogue systems for online shopping. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press.