AutoEvolve: Automatically Evolving Queries for Applicable and Scalable Retrieval-Augmented Generation Benchmarking

Dingchu Zhang[1,3*], **Xiaowen Zhang**[3*], **Yue Fei**[3], **Renjun Hu**[2],
Xiaowen Yang[1], **Zhi Zhou**[1], **Baixuan Li**[3], **Yufeng Li**[1†], **Xing Shi**[3], **Wei Lin**[3]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, China
and School of Artificial Intelligence, Nanjing University, China
[2] East China Normal University, [3]Alibaba Cloud Computing, China
{zhangdc,yangxw,zhouz,liyf}@lamda.nju.edu.cn, rjhu@dase.ecnu.edu.cn
{zxw320697,luxun.fy,libaixuan.lbx,shubao.sx,weilin.lw}@alibaba-inc.com

Abstract

Retrieval-augmented generation (RAG) enables large language models (LLMs) to address queries beyond their internal knowledge by integrating domain knowledge in specialized corpus, which necessitates the generation of benchmarks on specific corpus to evaluate RAG systems. However, existing automated generation methods exhibit *Weak Applicability* and *Weak Scalability*. *Weak Applicability* refers to the reliance on metadata from specific corpora for query generation, constraining applicability to other corpora. *Weak Scalability* is characterized by fixed query content after generation, unable to dynamically increase difficulty, limiting scalability of the query. To overcome these issues, we propose **AutoEvolve**, an applicable approach for dynamically evolving queries to construct scalable RAG benchmarks. Our approach is grounded in three key innovations: (i) a corpus-agnostic method for constructing the universal entity-document graph; (ii) a suite of evolution operations designed to dynamically update queries; and (iii) a difficulty-guided metric that directs query evolution process. Through experiments on three generated benchmarks, we demonstrate that AutoEvolve evolves queries that are significantly more challenging, paving the way for more applicable and scalable RAG evaluations.

1 Introduction

Retrieval-augmented generation (RAG) systems are attracting growing attention (Gao et al., 2023; Asai et al., 2024; Yu et al., 2024) due to their capacity to enable large language models (LLMs) to answer queries beyond their internal knowledge by integrating domain knowledge in specialized corpus. Therefore, it is necessary to automatically construct an effective benchmark on a specific corpus rather than relying on generic benchmarks to evaluate the RAG system.

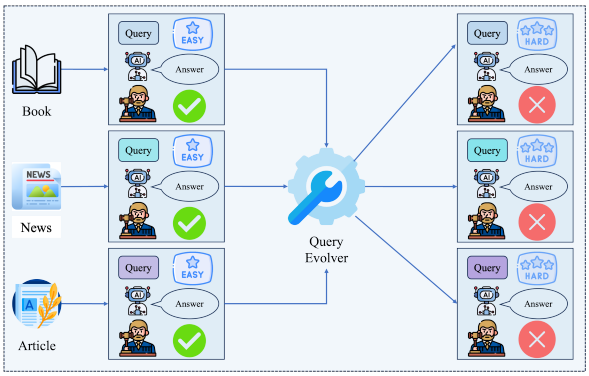

Figure 1: Evolving queries on different corpora to increase their difficulty.

However, existing methods (Tang and Yang, 2024; Hou et al., 2024) for automatically generating benchmarks face issues of *Weak Applicability* and *Weak Scalability*. *Weak Applicability* pertains to reliance on metadata derived from specific corpora, which lacks the ability to apply to other corpora. Specifically, (Yang et al., 2018; Ho et al., 2020) utilize link attributes from Wikipedia corpus to generate multi-hop queries, whereas (Zhu et al., 2024) leverages schema extracted from specific domains to assist in generating a diverse array of query types. *Weak Scalability* refers to the fixed query content after generation, which cannot dynamically increase the difficulty, thereby limiting scalability of the query.

To overcome these issues, we propose AutoEvolve, an applicable approach for automatically evolving queries to construct scalable RAG benchmarks, as illustrated in Figure 1. Specifically, to alleviate the issue of *Weak Applicability*, we utilize a corpus-agnostic method to construct a universal entity-document relationship graph to establish the relationships among documents by extracting consistent entities from documents. Simultaneously, we propose an evolutionary framework to dynam-

[*]These authors contributed equally.
[†]Correspondence.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 7624–7639
November 4-9, 2025

ically enhance the difficulty of queries based on the graph, thereby addressing the issue of *Weak Scalability*. Specifically, we introduce a suite of evolution operations to dynamically update the existing queries and present a difficulty-guided metric without the intervention of LLMs for assessing query difficulty to guide the direction of evolution in the framework.

In experiments, we demonstrate the applicability of AutoEvolve by generating benchmarks from three corpora and evaluating seven retrievers and LLMs. Furthermore, we compare the performance of retrievers before and after evolution, as well as the overall performance of the RAG system, demonstrating the scalability of AutoEvolve. Specifically, the evolution of the queries leads to a HIT@6 decline of 21.4% for bge-m3 on Booksum-E and 17.3% on MultiHopRAG-E. For MultiHopRAG-E, Recall-kp for gpt-4o and qwen2.5-7b-ins decreases by 12.4% and 14.6%, respectively. These findings indicate that AutoEvolve can dynamically increase the difficulty of queries across different corpora, opening the door for more transferable and scalable RAG evaluations.

Overall, our contributions are as follows:

- We propose AutoEvolve, where the generated queries are based on the construction of a universal entity-document relationship graph, enabling it to adapt to different corpus styles.
- AutoEvolve can identify relevant information in the corpus to evolve queries, ensuring their difficulty exceeds a predefined threshold, thereby enabling controllable difficulty in query generation.
- We generate three benchmarks on three different corpora and evaluate multiple retrievers and LLMs. Our experiment results show that the evolved queries are more challenging across multiple benchmarks, paving the way for more transferable and scalable RAG evaluations.

2 Related Work

RAG Evaluation: As RAG systems grow in popularity, various RAG benchmarking datasets and evaluation tools have emerged. For instance, RGB (Chen et al., 2024b) and RECALL (Liu et al., 2023) focus on assessing LLMs' response generation for RAG systems in conditions with noisy, integrative, and counterfactual queries, yet they primarily target the generation aspect and overlook retrieval accuracy. Automated evaluation tools like ARES (Saad-Falcon et al., 2024), RAGAS (Es et al., 2024), and RAGChecker (Ru et al., 2024) employ LLMs to evaluate RAG generation quality but lack associated benchmarking datasets. To address these gaps, our work offers a method to construct scalable benchmarks, including queries, relevant documents, and ground truth, thus complementing existing RAG evaluations.

Corpus	Type	Entry Count	Avg. Tokens
CNN Daily	News Summary	600	1906.5
Booksum	Book Summary	3542	513.9
MultiHopRAG	Article	609	2585.0

Table 1: Details of the knowledge base in three corpora. Corpora from different domains exhibit variations in entries, average token counts, and style type.

RAG Benchmarks: Apart from the RAG evaluation, numerous benchmarking datasets exist for information retrieval system assessments. For instance, the SciFact dataset includes scientific claims with evidence from abstracts (Wadden et al., 2020), but these claims are single-hop statements, differing from the multi-chunk queries in this paper. The HoVer dataset involves claims requiring extraction and reasoning across multiple Wikipedia articles (Jiang et al., 2020). However, HoVer focuses solely on classifying claims as supported or unsupported, omitting language model generation evaluation. Additionally, benchmarks like HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020) address QA from multiple document sources, yet they are exclusively Wikipedia-derived, limiting their construction methodology's applicability to other corpora. Recent studies (Guinet et al., 2024) on automated benchmark generation, such as MultiHopRAG (Tang and Yang, 2024) and RAGEval (Zhu et al., 2024), aim to create benchmarks without human input. However, these benchmarks lack metrics to evaluate query difficulty and have predetermined query types, restricting adaptive difficulty enhancement for new queries based on existing ones.

3 Method

3.1 Preliminary

In a RAG application, we leverage an external corpus, denoted as $\mathcal{D}$, which consists of multiple docu-

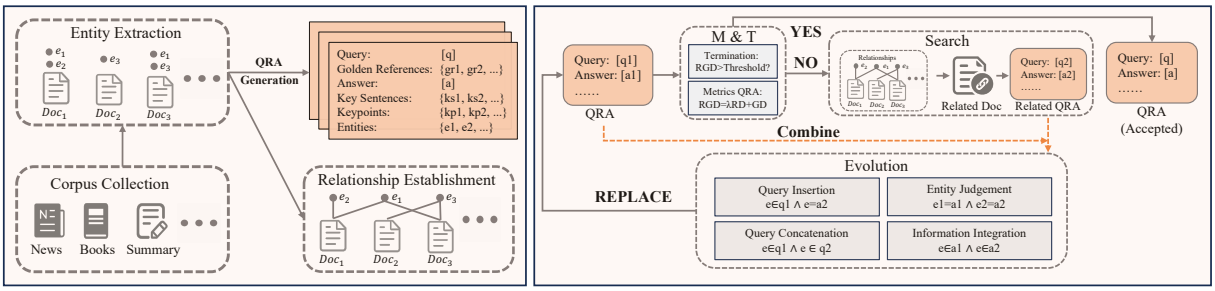

Figure 2: An overview of the AutoEvolve. AutoEvolve links documents by consistent entities and generates QRAs. It evaluates each QRA's difficulty score and, if necessary, seeks related documents and QRAs to evolve the origin QRA into a more challenging one, replacing the simpler one, until the difficulty score meets the requirements.

ments and functions as knowledge base. Each document within this corpus, represented as $d_i \in \mathcal{D}$, is segmented into a collection of chunks. These chunks are subsequently transformed into vector representations using an embedding model and stored in an embedding database. Upon receiving user query q, the system retrieves the top-K chunks that most closely match the query. These retrieved chunks form the retrieval set for query q, denoted as $\hat{\mathcal{R}}_q = \{\hat{r}_1, \hat{r}_2, ..., \hat{r}_K\}$. The retrieved chunks, in conjunction with the query and a prompt, are then input into an LLM to generate a final response re, following the format: LLM$(prompt, q, \hat{\mathcal{R}}_q) \rightarrow re$.

3.2 Entity-document Graph Construction

The construction of the entity-document relationship graph involves establishing brief relationships between documents through entities, which aids in generating queries across the documents and increases the difficulty of the queries. The construction process is systematically divided into three steps: corpus collection, entity extraction, and relationship graph construction.

Corpus Collection: We select MultiHopRAG (Tang and Yang, 2024), Booksum (Kryściński et al., 2022), and CNN Daily (See et al., 2017; Hermann et al., 2015) as our corpora for generating queries to evaluate the performance of the RAG system. Specifically, we utilize the entirety of MultiHopRAG as our corpus. For BookSum, to prevent content duplication within the corpus, we select a summary of each chapter from a curated selection of storybooks. In the case of CNN Daily, we extract a selection of long articles to constitute our corpus. Relevant information about the corpus is provided in Table 1.

Entity Extraction: Upon completing the corpus collection, we utilize the gpt-4o-mini to extract entities from each document, specifically concentrating on nouns that appear within the documents, including locations, people, events and so on. Through these extracted entities, we can establish connections between documents using the same entities, thereby formulating cross-document queries. The prompt template is shown in Appendix A.2.

Relationship Graph Construction: After entity extraction, we establish document relationships based on the identified entities. To address potential modifiers introduced during LLM-based entity extraction, we use Jaccard similarity to measure entity similarity and link documents accordingly. For each extracted entity e, we gather its candidate references, denoted as $\mathcal{R}_e$.

3.3 QRA Generation

To enhance the applicability of the method, we generate Query-Reference-Answer (QRA) based on the agnostic constructed relationship graph. Specifically, the QRA generation process can be summarized as follows: $e \rightarrow \mathcal{R} \rightarrow q \rightarrow \mathcal{GR}_q \rightarrow (a_q, \mathcal{KS}_q) \rightarrow \mathcal{KP}$, where these elements are denoted by QRA$= \{q, \mathcal{GR}_q, a_q, \mathcal{KS}_q, \mathcal{KP}_q\}$. This sequence outlines the process of data generation, which begins with the selection of an entity e within a chunk. Subsequently, we locate all chunks that contain this entity, represented as candidate references $\mathcal{R}$. From this set, we select a subset of references and utilize the relevant information regarding the entity to construct a query q that is centered on e. To assess references associated with this query, both q and $\mathcal{R}$ are input into an LLM, which outputs the golden references corresponding to q, denoted by $\mathcal{GR}_q = \{gr_1, gr_2, ..., gr_N\}$.

Upon identifying golden references, query q and golden references $\mathcal{GR}_q$ are fed into the LLM to generate the answer a_q and the key sentences utilized to assess the difficulty score, represented as $\mathcal{KS}_q = \{ks_1, ks_2, ..., ks_m\}$, where every sentence in $\mathcal{KS}_q$ is extracted from one of the references in the golden references, i.e. $ks_j \in gr_k$. Finally, we synthesize the extracted sentences $\mathcal{KS}_q$ by combing information from the answer a_q and golden references to form keypoints $\mathcal{KP}_q$ for the query q, which are utilized to evaluate the quality of the response in the section 4.

3.4 QRA Evolution Framework

We propose the QRA evolution framework to dynamically update queries that encompass multiple documents by combining them with relevant information from related documents, thereby enhancing the difficulty. This framework is systematically divided into five steps: Initialization, Metrics, Search, Evolution and Termination.

Initialization: We initialize several QRAs as the foundation of our framework. Specifically, we select an entity e and its candidate references $\mathcal{R}_e$ that contain this entity. Subsequently, based on the chosen entity and the selected document, we generate an entity-centered query and the corresponding QRA. For the initialization process, we define three common types of QRAs as the initialization. Detailed descriptions of the initialized QRA types can be found in Appendix B.

Metrics: To measure the difficulty of the generated QRA and guide the direction of subsequent evolution, we propose a metric to assess the difficulty of the generated QRA. This metric is divided into two distinct components: retrieval difficulty and generation difficulty. For retrieval difficulty RD, our objective is to evaluate the difficulty associated with retrieving golden references in response to the query q, denoted as follows:

$$RD(QRA) = \max_{r \in \mathcal{R}}(sim(q, r)) - \frac{1}{|\mathcal{GR}_q|} \sum_{i=1}^{|\mathcal{GR}_q|} sim(q, gr_i) \quad (1)$$

which means the gap between the maximum similarity of the query with the candidate references $\mathcal{R}$ and the average similarity of the query with the golden references $\mathcal{GR}$. Candidate references refer to all relevant documents that contain an entity appearing in either the query or the answer within the QRA. Furthermore, we employ one embedding model to assess the similarity and utilize the candidate references rather than the entire corpus to significantly decrease resource consumption. On the other hand, in terms of generation difficulty GD, our objective is to assess the difficulty associated with synthesizing information from multiple documents to adequately answer the queries, as illustrated by:

$$GD(QRA) = H(\mathcal{KS}_q) = \sum_{i=1}^{|\mathcal{GR}_q|} p_i log \frac{1}{p_i} \quad (2)$$

where p_i represents the distribution of sentences from $\mathcal{KS}_q$ in $\mathcal{GR}_q$. For example, if $\mathcal{KS}_q$ comprises three sentences and two of them are extracted from gr_i, then $p_i = \frac{2}{3}$. Ultimately, we define the Retrieval Generation Difficulty (RGD) as a metric to assess the difficulty of QRA, following the format:

$$RGD(QRA) = GD + \lambda RD \quad (3)$$

Search: When the difficulty of the generated QRA does not exceed a specific threshold δ, we search for documents containing relevant information to create cross-document queries, thereby increasing the difficulty of the QRA. Specifically, for each QRA, we first identify the entities present in either the query or the answer. Subsequently, for each extracted entity e, we locate the candidate references that contain this entity, which ensures that the identified documents are relevant and facilitates the generation of QRA that integrates information across these documents.

Evolution: Upon identifying relevant entities and candidate references, we randomly select a subset of these documents to extract relevant information. Specifically, for each document, we generate an entity-centric QRA to serve as additional information that aids in evolving the original QRA. We propose four types of evolutionary operations and their conditions to merge the information from two QRAs to generate a more difficult query. These types are specifically categorized as query insertion, entity judgment, query concatenation, and information integration.

(1) $Query\ insertion$. The evolution condition for query insertion is that an entity serves

	CNN Daily-E			Booksum-E			MultiHopRAG-E		
Embedding	MRR@5	MAP@5	HIT@5	MRR@5	MAP@5	HIT@5	MRR@5	MAP@5	HIT@5
bge-large-en-v1.5	50.21	63.52	78.21	40.55	48.05	60.74	40.45	48.07	61.09
bge-m3	**52.51**	**67.11**	**80.63**	**43.22**	**52.19**	**65.01**	**42.96**	**51.18**	**64.30**
e5-base-v2	50.15	64.11	79.32	38.91	46.18	58.62	38.81	46.09	60.34
gte-large-en-v1.5	49.34	62.42	76.53	35.26	41.67	54.49	37.87	44.99	58.71
instructor-large	47.80	60.29	74.22	37.54	44.16	55.89	37.71	44.22	57.01
jina-embeddings-v2-base-en	45.60	56.58	70.89	36.14	42.34	53.86	36.15	42.22	55.13
llm-embedder	42.01	51.94	66.73	24.75	28.21	38.89	27.68	32.10	43.89

Table 2: Retrieval performance of different embedding models on three benchmarks.

	CNN Daily-E			Booksum-E			MultiHopRAG-E		
Generator	Rouge-l	Fscore-c	Recall-kp	Rouge-l	Fscore-c	Recall-kp	Rouge-l	Fscore-c	Recall-kp
GPT-4o-mini	40.71	62.00	67.22	35.24	58.30	61.55	37.34	55.70	61.97
GPT-4o	**47.60**	**67.10**	68.68	**41.97**	**61.70**	**64.01**	**42.71**	**59.00**	63.79
Qwen-max	33.96	57.70	**71.60**	26.63	53.70	61.95	29.48	49.60	**64.68**
Qwen-2.5-7b-ins	39.34	55.50	60.79	34.15	50.50	53.62	35.78	48.20	55.31
Glm3-6b	28.55	30.00	34.41	21.96	23.30	25.07	26.85	24.60	27.73
Mistral-7b-ins-v0.2	38.70	46.90	50.66	34.77	43.80	48.85	36.63	41.90	45.79
Llama-3.1-8b	30.71	51.00	55.03	28.69	48.40	51.76	28.67	43.90	48.65

Table 3: Generation performance of different LLMs on three datasets with retriever bge-m3.

as the answer to one query while appearing in another query, which is formalized as $e = a_1 \wedge e \in q_2$. This evolution mechanism serves to substitute an entity in the query with a clause, thereby transforming it into a multi-hop query and increasing its difficulty.

(2) *Entity judgment*. Entity judgment requires that the answers to two queries belong to the same category of entities, and then constructs queries to determine whether they are equivalent. This is formalized as $e_1 = a_1 \wedge e_2 = a_2$ and e_1 may be equal to e_2. In addition to requiring that the retriever retrieves all documents related to the queries, LLM also needs to determine whether the answers are semantically equivalent. This represents a challenging category of queries for RAG systems.

(3) *Query concatenation*. Query concatenation refers to connecting two related queries, formalized as $e \in q_1 \wedge e \in q_2$. The goal is to transform a single query into a compound query, which evaluates the completeness of the retriever's retrieval capabilities and the comprehensiveness of the model's responses.

(4) *Information integration*. Information integration involves combining related information from different documents and formulating a query that spans across these documents, formalized as $e \in a_1 \wedge e \in a_2$. The objective is to dynamically synthesize related information from various documents to formulate a query that necessitates the integration of information from multiple documents for its resolution.

Following the above process, we check whether the evolved queries necessitate the integration of multiple documents for their resolution and eliminate any unqualified QRAs. Additionally, to mitigate resource consumption, we only retrain the two most challenging QRAs in the queue for subsequent search and evolution whenever multiple QRAs are generated, discarding the remainder. Examples of four types of evolution are available in the Appendix A.1.

Termination: After evolving the QRAs, we sequentially repeat the aforementioned steps until the average RGD of the current QRAs surpasses the established threshold and attains their maximum. Ultimately, we output the generated most challenging QRA. The overall framework of AutoEolve is shown in Figure 2.

4 Experiments

4.1 Corpus and Models

We utilize corpora MultiHopRAG (Tang and Yang, 2024), CNN Daily (See et al., 2017; Hermann et al.,

		bge-m3			bge-m3 + GPT-4o			bge-m3 + Qwen2.5-7b-ins		
Benchmark	Score	HIT	MRR	MAP	Rouge-l	Fscore-c	Recall-kp	Rouge-l	Fscore-c	Recall-kp
CNN Daily-E	*RD*	**-0.3540**	**-0.5400**	**-0.5491**	-0.1034	-0.0499	**-0.0578**	-0.0805	-0.0825	-0.0280
	GD	-0.2305	-0.0233	-0.0095	-0.1493	-0.1968	-0.1233	-0.1516	-0.1433	**-0.1950**
	RGD	-0.3420	-0.3233	-0.3123	**-0.1845**	**-0.1996**	-0.1529	**-0.1883**	**-0.1669**	-0.1784
BookSum-E	*RD*	**-0.5024**	**-0.5380**	**-0.5506**	-0.1084	-0.2353	**-0.2843**	-0.1466	-0.1644	-0.2565
	GD	-0.2412	-0.1242	-0.1251	**-0.3114**	-0.2434	-0.2151	-0.2185	**-0.2119**	-0.3204
	RGD	-0.3477	-0.3620	-0.3619	-0.2992	**-0.2823**	-0.2799	**-0.2372**	-0.1952	**-0.3511**
MultiHopRAG-E	*RD*	-0.3221	**-0.3125**	**-0.3279**	-0.0745	-0.0870	**-0.2171**	-0.0668	-0.0721	-0.1294
	GD	-0.2108	-0.0518	-0.0878	**-0.3090**	-0.1643	-0.1937	**-0.2290**	-0.2236	-0.1692
	RGD	**-0.3314**	-0.2375	-0.2598	-0.2818	**-0.1928**	-0.2148	-0.1886	**-0.2391**	**-0.2088**

Table 4: **Spearman rank correlation**↓ between the generated metrics and the retrieved metrics with the defined difficulty scores across different LLMs and benchmarks.

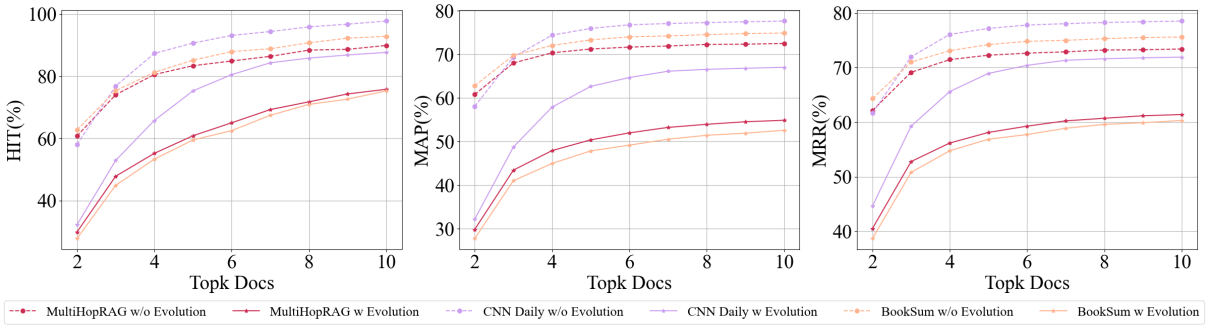

Figure 3: The relationship between retrieval metrics and the number of retrieved documents about retriever bge-m3 with or without evolution.

2015) and Booksum (Kryściński et al., 2022) to generate 694, 758, and 821 evolved QRAs respectively, named MultiHopRAG-E, CNN Daily-E, and BookSum-E. The distribution of various evolved queries and the number of evolution iterations in the generated benchmarks are shown in the Appendix F and Appendix H, respectively. In these experiments, we test a variety of embedding models as our retriever, including bge-large-en-v1.5 (Xiao et al., 2023), bge-m3 (Chen et al., 2024a), e5-base-v2 (Wang et al., 2022), gte-large-en-v1.5 (Zhang et al., 2024; Li et al., 2023), instructor-large (Su et al., 2023), jina-embeddings-v2-base-en (Günther et al., 2023) and llm-embedder (Zhang et al., 2023). Furthermore, we evaluate several LLMs as our generator, including llama-3.1-8b (Touvron et al., 2023), mistral-7b-instruction-v0.2 (Jiang et al., 2023), glm3-6b (GLM et al., 2024), qwen-2.5-7b-instruction, qwen-max (Yang et al., 2024), gpt-4o-mini and gpt-4o (Achiam et al., 2023), utilizing the top-performing embedding model bge-m3 as the retriever for the generation process.

4.2 Evaluation Metrics

A RAG system handling queries can be assessed from two key aspects: retrieval evaluation and response evaluation. For the Retrieval Evaluation, we utilize several evaluation metrics, including Mean Average Precision at K (MAP@K), Mean Reciprocal Rank at K (MRR@K), and Hit Rate at K (HIT@K). For the Response Evaluation, we utilize the Rouge-l, which is based on the longest common subsequence, to assess the coverage of word frequency. Additionally, we leverage the Fscore-c of the claims extracted by LLM to evaluate the consistency between the response and the answer, as referenced in RAGChecker (Ru et al., 2024). Finally, we utilize the Recall-kp, which quantifies the recall of keypoints in the response, to evaluate the completeness of the response.

4.3 Implementation Details

We divide documents in the corpus into 512 tokens to build document relationships. Using gpt-4o-mini, we extract entities from each chunk and link documents through shared entities. Noticing

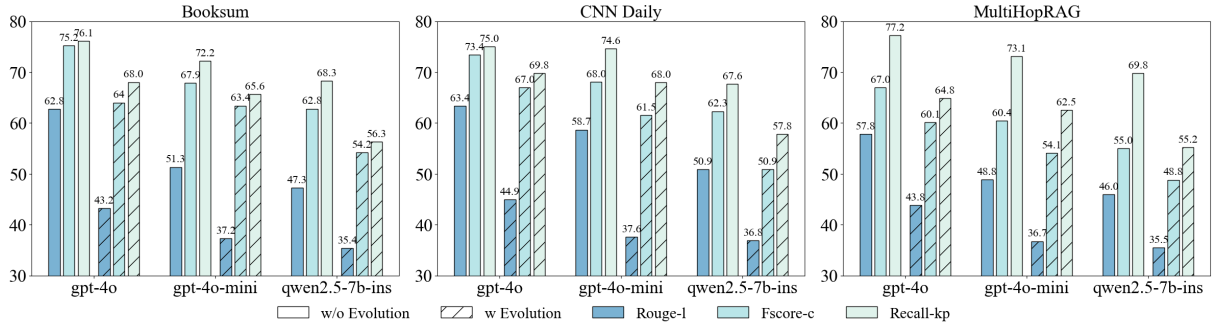

Figure 4: The overall performance of RAG systems across three models on three metrics with or without evolution when retrieving 6 documents.

Metric	MultiHopRAG-E	CNN Daily-E	Booksum-E
Context Relevancy	94.73/100.0/96.43	94.44/90.91/91.67	100.0/100.0/100.0
Answer Relevancy	78.95/88.24/85.71	94.44/100.0/97.22	86.96/90.00/91.67
Answer Correctness	78.95/82.35/85.71	94.44/86.36/88.89	86.96/95.54/94.44
Key Sentence Completeness	94.73/88.24/92.86	100.0/100.0/100.0	82.61/81.25/80.56
Keypoint Relevancy	67.89/70.58/67.29	72.22/81.82/77.56	73.91/63.25/66.67
Keypoint Correctness	73.68/82.35/77.43	83.33/95.45/83.11	82.85/80.95/83.11

Table 5: Human scoring (%) of QRA quality across different benchmarks by three distinct human groups.

that RD is much smaller than GD, we set λ to 20 to balance them. We also establish a termination threshold δ of 1.2 and a queue size of 2. During the RAG system evaluation, we embed queries with the same model and set K to 5 to retrieve the most relevant documents based on cosine similarity. Additionally, we set the generation model's temperature to 0 to generate responses using the retrieved documents.

4.4 Main Results

We create three benchmarks from different corpora to evaluate seven retrievers and seven LLMs, highlighting AutoEvolve's applicability. Table 2 shows bge-m3 outperforming other retrievers, with hit rate advantages of 3.21% on MultiHopRAG-E, 1.31% on CNN Daily-E, and 4.27% on BookSum-E. Table 3 uses bge-m3 for LLM assessment, revealing qwen2.5-7b-instruct as the leading open-source model, surpassing llama3.1 in Recall-kp across all benchmarks. For closed-source models, gpt-4o generally leads, though qwen-max occasionally scores higher in Recall-kp. When the RAG system's performance is below the hit score, its efficiency depends heavily on the generator's capabilities. The hit score is tied to the system's performance ceiling, with qwen-max slightly exceeding it on MultiHopRAG-E, while gpt-4o falls short on BookSum-E. Identifying vulnerabilities in the RAG system is crucial for improving overall performance.

AutoEvolve can evolve queries based on conditions to increase their difficulty. We demonstrated the scalability of AutoEvolve by comparing retriever and LLM performance before and after evolution. Figure 3 shows retrieval performance initially improves with fewer documents but slows when exceeding 4 (6) documents, with evolution causing significant declines in hit metrics—dropping 21.4%, 10.05%, and 17.13% on Booksum-E, CNN Daily-E, and MultiHopRAG-E benchmarks, respectively. Figure 4 highlights performance drops among three LLMs following evolution, with the bge-m3 retrieving the top 6 documents for consistency. On MultiHopRAG-E, Recall-kp decreased by 12.4% for gpt-4o, 10.6% for gpt-4o-mini, and 14.6% for qwen2.5-7b-ins. The smaller model was especially impacted, showing declines in Recall-kp by 8.1% and 6.6% with gpt-4o and gpt-4o-mini on Booksum-E, while qwen2.5-7b-ins decreased by 12%.

The defined difficulty metric can reflect the difficulty of a query, thereby guiding the evolution

Evolution Type	Rouge-l	Fscore-c	Recall-kp
Query Insertion	41.71	57.86	63.56
Entity Judgement	50.57	64.36	69.10
Query Concatenation	38.90	56.81	61.30
Information Integration	35.62	55.09	61.56

Table 6: The generation performance of gpt-4o on different evolution types in the MultiHopRAG-E.

process. We demonstrate the functionality of the metrics guidance by exploring the relationship between the defined difficulty metrics and the performance of retrievers and LLMs. Table 4 shows the Spearman correlation coefficient between difficulty metrics and performance. A smaller coefficient indicates that increasing difficulty makes performance decline easier, validating the use of evolution. From RD's perspective, difficulty shows a strong negative correlation with retrieval metrics, affecting retriever and RAG system performance. Similarly, GD reveals a strong negative correlation with generation metrics, increasing integration difficulty for LLMs. Combining RD and GD into RGD results in a strong negative correlation with RAG system performance, demonstrating effective integration.

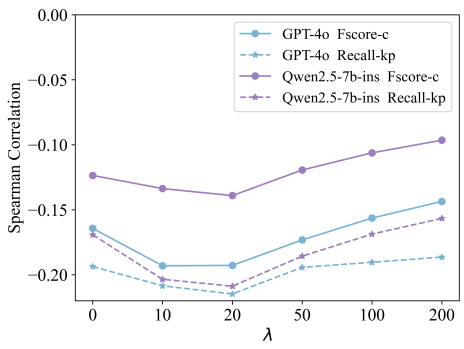

Figure 5: The spearman correlation with different hyperparameter λ on benchmark MultiHopRAG-E.

5 Discussion

5.1 Quality Assessment

Experiment Setup To assess the quality of the generated benchmarks, 240 samples, representing 10% of the entire dataset, are scored. Six human evaluators participate, providing each sample with three independent scores. The evaluators work in three groups, each scoring 120 randomized samples. Scoring focused on six dimensions rated 0 or 1: context relevancy, answer relevancy, answer correctness, key sentence completeness, keypoint relevancy, and keypoint correctness. Details of the evaluation metrics are provided in Appendix G.

Results Analysis From Table 5, the quality of the generated QRA is generally good across the six dimensions evaluated by humans. However, when queries require integrating extensive contextual information to formulate a response, the proliferation of content within the answers can result in diminished relevance and accuracy of the extracted keypoints. Therefore, enhancing the model's ability to comprehend extended contexts is crucial for improving data generation quality.

5.2 Performance of Each Evolution Type

Table 6 presents the performance of the gpt-4o with the bge-m3 retriever across four types of queries. The analysis reveals that the model exhibits diminished proficiency in addressing queries categorized under Query Concatenation and Information Integration. In contrast, it demonstrates enhanced capability in resolving queries pertaining to Entity Judgement.

5.3 The Effect of Hyperparameter λ

Figure 5 demonstrates the effect of varying hyperparameter λ on metrics Fscore-c and Recall-kp for the MultiHopRAG-E benchmark. To facilitate analysis, we employ the Spearman correlation coefficient to assess the relationship between the overall performance and the hyperparameter. Notably, when the hyperparameter λ is within the range of 10 to 50, a strong negative correlation is observed between the difficulty score and the overall performance of the RAG system.

6 Conclusion

In this paper, we investigate the automatic generation of effective benchmarks for evaluating RAG systems on a specific corpus. We observe that existing methods suffer from two issues: *Weak Applicability* and *Weak Scalability*. To address these issues, we propose AutoEvolve, an applicable approach for automatically evolving queries to construct scalable RAG benchmarks. For *Weak Applicability*, we propose a corpus-agnostic method for constructing the universal entity-document relationship graph. For *Weak Scalability*, we utilize the automatic evolution framework to dynamically enhance the difficulty of queries based on the graph. Through

experiments on three benchmarks that AutoEvolve generates from three different corpora, we demonstrate that evolved queries are more challenging, opening the door for more transferable and scalable RAG evaluations.

Limitations

This paper studies the automatic generation of evaluation benchmarks for RAG on specific corpora. Current methods face the issues of *Weak Applicability* and *Weak Scalability*. Therefore, we propose AutoEvolve to address these two issues. Specifically, we construct a universal entity-document relationship graph that captures the entities shared among documents to build the relationships. Additionally, we propose an evolutionary framework to dynamically enhance the difficulty of queries based on the graph. Our work gives some instructions on the automated generation of evaluation data. At the same time, there is still much room for exploration in this approach. We hope this work will attract attention in the future to explore more precise query difficulty metrics and a greater variety of evolving query types.

Acknowledgement

This research was supported by the Jiangsu Science Foundation (BG2024036, BK20243012), Natural Science Foundation of China (62576162,62576174), and the Fundamental Research Funds for the Central Universities (022114380023).

References

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

Akari Asai, Zexuan Zhong, Danqi Chen, Pang Wei Koh, Luke Zettlemoyer, Hannaneh Hajishirzi, and Wen-tau Yih. 2024. Reliable, adaptable, and attributable language models with retrieval. *arXiv preprint arXiv:2403.03187*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024b. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the 45th AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

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.

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*.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, and 1 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Gauthier Guinet, Behrooz Omidvar-Tehrani, Anoop Deoras, and Laurent Callot. 2024. Automated evaluation of retrieval-augmented language models with task-specific exam generation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 16773–16801.

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, and 1 others. 2023. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents. *arXiv preprint arXiv:2310.19923*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, volume 28.

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.

Yufang Hou, Alessandra Pascale, Javier Carnerero-Cano, Tigran Tchrakian, Radu Marinescu, Elizabeth Daly, Inkit Padhi, and Prasanna Sattigeri. 2024. Wikicontradict: A benchmark for evaluating llms on real-world knowledge conflicts from wikipedia. *arXiv preprint arXiv:2406.13805*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. 2020.

HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Proceedings of the 27th Conference on Empirical Methods in Natural Language Processing*.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. Booksum: A collection of datasets for long-form narrative summarization. In *Findings of the Proceedings of the 29th Conference on Empirical Methods in Natural Language Processing*, pages 6536–6558.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Yi Liu, Lianzhe Huang, Shicheng Li, Sishuo Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Recall: A benchmark for llms robustness against external counterfactual knowledge. *arXiv preprint arXiv:2311.08147*.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, and 1 others. 2024. Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. Ares: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 338–354.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of Proceedings of the 61th Association for Computational Linguistics*, pages 1102–1121.

Yixuan Tang and Yi Yang. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*.

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*.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 27th Conference on Empirical Methods in Natural Language Processing*, pages 7534–7550.

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

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

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

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 25th Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of retrieval-augmented generation: A survey. *arXiv preprint arXiv:2405.07437*.

Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023. Retrieve anything to augment large language models. *arXiv preprint arXiv:2310.07554*.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, and 1 others. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. *arXiv preprint arXiv:2407.19669*.

Kunlun Zhu, Yifan Luo, Dingling Xu, Ruobing Wang, Shi Yu, Shuo Wang, Yukun Yan, Zhenghao Liu, Xu Han, Zhiyuan Liu, and 1 others. 2024. Rageval: Scenario specific rag evaluation dataset generation framework. *arXiv preprint arXiv:2408.01262*.

A Prompt Templates

A.1 Prompt Template of Four Types of Evolution

We present four types of evolution prompt templates. Specifically, Figure 8 shows the prompt template for query insertion, Figure 9 illustrates the template for information integration, Figure 10 demonstrates the template for entity judgment, and

Figure 11 presents the template for query concatenation. Additionally, within each prompt template, we provide two examples to illustrate the specific details of our evolution.

A.2 Prompt Template of Entities Extraction

In Figure 6, we present the prompt template used by gpt-4o-mini to extract all entities from the text.

B Examples of the initial QRA types.

In Table 8, we present three types of initialization for QRAs. We categorize the queries into three types: entity query, details query, and statement query. An entity query is a question about a specific entity, which is the answer to the query. A details query focuses on a specific detail or aspect of an entity. A statement query, on the other hand, involves asking a summarizing or conclusive question about an entity.

C Details of QRA.

In Figure 7, we present an example of QRA, which includes six components: query, golden references, answer, key sentences, keypoints, and entities. The query is a request for information about a specific entity. Golden references are documents related to the query that contains the answer. The answer is the response to the query, derived from the golden references. Key sentences are reference sentences extracted from the golden references, used to define the generation difficulty. Keypoints serve as crucial criteria for evaluating the RAG system's response. Entities are mentioned in either the query or the answer, used to help search for relevant documents to evolve the query.

D Details of Methods.

In Algorithm 1, we present the automatic evolution framework. We use a queue to store all QRAs. For each QRA in the queue, we find all associated entities, and through these entities, we identify candidate references. Next, we randomly select one document from the candidate references and generate an associated QRA. This new QRA is then evolved with the original QRA to create a more challenging QRA, which is added back into the queue. This process is repeated num_e times. Upon completion, we only retain the qs most difficult QRAs in the queue. The aforementioned steps are then repeated until the difficulty scores of the QRAs in the queue converge. After convergence, we extract the most challenging QRA from the queue. If the difficulty score of this QRA exceeds a predefined threshold, it is outputted; otherwise, we start over with a different initial QRA.

Number of Evolution	0	1	2	>=3	All
MultiHopRAG-E	116	561	12	5	694
CNN Daily-E	108	626	24	0	758
BookSum-E	124	676	18	3	821

Table 7: Distribution of the number of evolutions for each generated benchmark.

E Distribution of RGD before and after Evolution

To demonstrate the distribution of RGD before and after evolution, we randomly sample 400 QRAs from each of three different corpora, both before and after evolution, and present the distribution of their RGD. In Figure 13, it can be observed that the RGD of most pre-evolution QRAs are concentrated between 0 and 0.5, whereas the RGD of post-evolution QRAs are mainly concentrated between 1.0 and 2.0. This demonstrates that the evolution process effectively increases the difficulty scores of QRAs, thereby generating a more challenging QRA to decrease the performance of RAG systems.

F Distribution of Generated Benchmarks

In Figure 12, we show the distribution of each type of evolved query across the three generated benchmarks. Entity Judgement has the highest evolution success rate, while Information Integration has the lowest. Additionally, in all three generated benchmarks, there are a few samples that reach the defined difficulty threshold δ without requiring evolution.

G Details of Dataset Evaluation Metrics

In Quality assessment, we evaluate the generated benchmarks based on six dimensions rated 0 or 1: context relevancy, answer relevancy, answer correctness, key sentence completeness, keypoint relevancy, and keypoint correctness. Specifically, context relevancy indicates whether all information in the query is present in the context. Answer relevancy refers to whether the answer depends on the context rather than the model's own knowledge. Answer correctness means the answer is accurately

Algorithm 1 AutoEvolve

Input: initial QRA qra_i, queue q, documents relationship graph drg, max queue size qs, threshold δ, score change ϵ, evolution quantity num_e.

Output: enhanced QRA qra_{en}.
Push qra_i into q.
$avg\_score = 0$
$new\_avg\_score = avg(RGD(qra \in q))$
while $new\_avg\_score > avg\_score + \epsilon$ **do**
  $avg\_score = avg(RGD(qra \in q))$.
  for $qra \in q$ **do**
    Get all entities es in qra.
    Get candidate documents cd by es from drg.
    for $i = 1$ **to** num_e **do**
      Randomly sample a document d from cd.
      Generate a new QRA qra_{new} from d by llm.
      $qra_e = Evolve(qra, qra_{new})$
      Push qra_e into q.
    end for
  end for
  Keep qs QRAs with the highest difficulty score in the q.
  $new\_avg\_score = avg(RGD(qra \in q))$.
end while
Get the QRA qra_{max} with the highest difficulty score in the q.
if $RGD(qra_{max}) > \delta$ **then**
  return qra_{max}.
else
  return None.
end if

provided based on the content of the context, addressing the question. Key sentence completeness signifies that the key sentences include all factual content relevant to the answer and are extracted from the contexts. Keypoint relevancy denotes whether the keypoints encompass all the factual content from the key sentences. Keypoint correctness means the key points contain all the crucial information necessary to answer the question.

H Distribution of Evolution Frequency

In Table 7, we display the number of evolution iterations for each constructed benchmark. Some queries do not undergo evolution because their answers require synthesizing information from multiple documents, indicating a relatively well-configured initialization. Furthermore, our analysis shows that for the specified difficulty threshold of δ=1.2, typically, a single evolution iteration is sufficient to surpass this threshold. If one wishes to further enhance the challenge of the benchmark or increase the number of evolution iterations, raising the difficulty threshold can be considered. However, due to limited funding, we do not adopt a high threshold in constructing the benchmark, but this does not negate the potential of AutoEvolve.

Extract all entities carefully from the given text to ensure none are overlooked, focusing on the most significant and distinctive aspects and no less than two words.
text: <text>
output: <all entities>

Figure 6: Prompt template for extracting all entities.

Query Type	Answer Type	Example
Entity Query	Short-Form	{Query: "What is the service provided by Amazon that many sellers find valuable for its price and promise to deliver two-day shipping?", Answer: "FBA", ...}
Details Query	Long-Form	{Query: "How did The Lions manage to secure a victory over the Raiders despite their critical mistakes and struggles in the red zone?", ...}
Statement Query	Long-Form	{Query: "Please tell me the impact of the expectation of long-term high interest rates on the Australian stock market and various industries.", ...}

Table 8: Examples of the initial QRA type.

{
Query: "Did the setting where the Ghost of Andrea served as a courtier also involve Hieronimo's coded discussion of injustice?"
Golden References: [**1.** "The Ghost of Andrea enters the scene to deliver a monologue and put forth the premises of the play. When we were alive, the Ghost states, he served as a courtier in the Spanish court ...",
2. "Two random Portuguese dudes question Hieronimo about the whereabouts of the Duke of Castile, but Hieronimo only gives evasive answers..."],
Answer: "Hieronimo talks in coded language about the injustice of the Spanish court. The Ghost of Andrea served as a courtier in the Spanish court during his lifetime. Therefore, both the setting where the Ghost of Andrea served as a courtier and Hieronimo's coded discussion of injustice involve the Spanish court.",
Key Sentences: [**1.** "When he was alive, the Ghost states, he served as a courtier in the Spanish court.",
2. "Hieronimo only gives evasive answers. But he does manage to talk in coded language about the injustice of the Spanish court."],
Keypoints: [**1**. "The Ghost of Andrea served as a courtier in the Spanish court during his lifetime.",
2. "Hieronimo discusses the injustice of the Spanish court using coded language, indicating his awareness of the court's corruption."],
Entities: [**1.** Spanish court.]
}

Figure 7: A QRA example of benchmark Booksum-E.

Assuming the answer to the question1 is the entity, please replace the entity in question2 with question1 to generate a new question, and rewrite and compress the newly generated question in an easily understandable way.

Here are some examples:
question1: "Where did a sharp drop for stocks occur, bringing them back to where they were in June?"
question2: "How has the increase in bond yields and other economic factors impacted Wall Street?"
entity: "Wall Street"
output: "How have the rise in bond yields and other economic factors affected the area where stocks experienced a significant decline, returning them to their June levels?”

question1: "Who was set up for a big play, but Goff's throw was just a bit behind him in the game between the Lions and the Raiders?"
question2: "How did Kalif Raymond contribute to the Lions' performance in the first quarter against the Raiders?"
entity: "Kalif Raymond"
output: "How did the player who was set up for a big play, but Goff's throw was just a bit behind him, contribute to the Lions' performance in the first quarter of the game against the Raiders?”

You actual task is:
question1: <question1>
question2: <question2>
entity: <entity>
output:

Figure 8: Prompt template of query insertion.

"A 'claim' is a statement or assertion made within a text expressing a belief, opinion, or fact. The 'claims' is the claims set extracted from different contexts. Please jointly generate a single question centered around the entity, requiring integrating claims extracted from different contexts as much as possible to answer."

Here are some examples:
claims:[
    ['The S&P 500 has experienced its fifth loss in the last six days, tumbling 1.5 per cent.', "The S&P 500 has lost 5.2 per cent in September, potentially making it the worst month of the year due to the Federal Reserve's decision to keep interest rates high."],
    ['Historically, the S&P 500 has dropped an average of roughly 10 per cent in the three months leading up to US government shutdowns.', 'Despite the initial drop, the S&P 500 has managed to hold up well during shutdowns, falling an average of just 0.3 per cent, before rebounding significantly afterward.']
],
entity: "S&P 500",
output: "How has the S&P 500's performance been affected by the Federal Reserve's decision to keep interest rates high and the historical trend of US government shutdowns?",

claims:[
    ["The Australian sharemarket, including healthcare and insurance sectors, experienced a decline due to the potential of higher interest rates for a longer period.", "Despite a negative lead from Wall Street and the latest inflation data, the Australian sharemarket remained resilient, indicating that markets are starting to adjust to the possibility of prolonged higher interest rates."],
    ["The Federal Reserve's decision to keep interest rates high has contributed to a 5.2 per cent loss for the S&P 500 in September.", "High interest rates set by the Federal Reserve are expected to continue for a long time.", "The S&P 500 is on track to have its worst month of the year due to the Federal Reserve's high interest rates."],
],
entity: "Interest rates",
output: "How have the high interest rates set by the Federal Reserve affected the Australian sharemarket and the S&P 500?",

You actual task is:
claims: <claims>
entity: <entity>
output:

Figure 9: Prompt template of information integration.

Assuming the answer to question1 is entity1 and the answer to question2 is entity2. Please generate a question that asks whether the answer to question1 is the answer to question2, and modify and compress the generated question to make it easier to understand. Note: Entity1 and entity2 cannot appear in the problem.

Here are some examples:
question1: "What type of product deals are included in Amazon's Cyber Monday sale, along with Echo, Fire TV, and Kindle deals?"
entity1:"Apple deals"
question2: "What section would you find the 10th generation 64GB model iPad for $349 on Amazon?"
entity2: "Apple deals"
output: "Along with deals on Echo, Fire TV, and Kindle products, will Amazon's Cyber Monday sale include the 10th generation 64GB iPad for $349?"

question1: "Where did a sharp drop in stocks occur, bringing them back to their status in June?"
entity1:"Wall Street"
question2: "Where is the impact of the highest bond market yields in over a decade and undercut prices for stocks and other investments being felt?"
entity2: "Wall Street"
output: "Did the sharp drop in stocks, returning them to June levels, and the impact of decade-high bond market yields occur in the same location?"

You actual task is:
question1: <question>
entity1: <entity>
question2: <question>
entity2: <entity>
output:

Figure 10: Prompt template of entity judgment.

Please merge question1 and question2 around the entity to generate a merged question, and then compress the merged question. The format of the merged question should be similar to that of question1 and question2.

Here are some examples:
question1:"Please tell me the effect of Wall Street's poor performance in September on the Australian markets."
question2: "Please tell me the factors, including high interest rates, that are currently affecting Wall Street."
entity: "Wall Street"
output: "Please tell me the effect of Wall Street's poor performance in September on the Australian markets and the factors, including high interest rates, that are currently affecting Wall Street."

question1:"Please tell me the contribution of Goff in extending the Lions' lead against the Raiders."
question2: "Please tell me the contributions of Goff in the first half of the Lions and Raiders game."
entity: "Goff"
output: "Please tell me the contribution of Goff in extending the Lions' lead against the Raiders and his contributions in the first half of the Lions and Raiders game."

You actual task is:
question1: <question>
question2: <question>
entity: <entity>
output:

Figure 11: Prompt template of query concatenation.

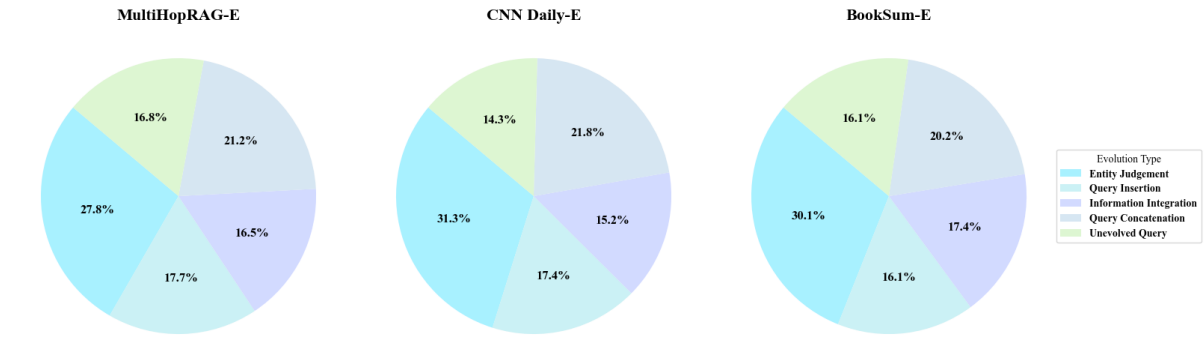

Figure 12: Distribution of different evolution types in the generated benchmarks.

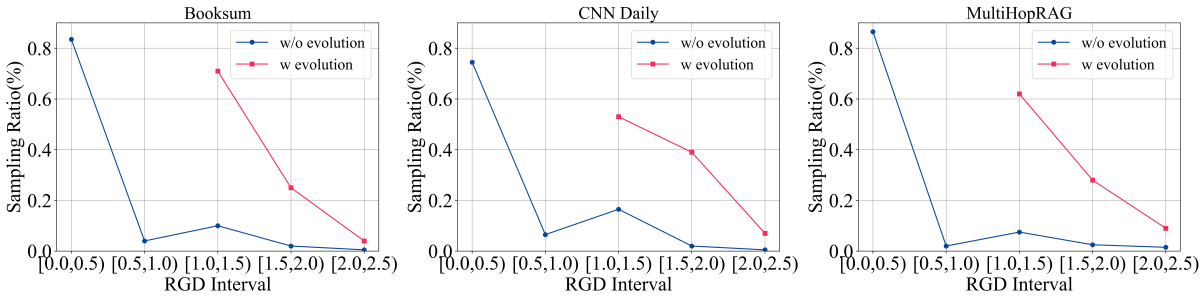

Figure 13: RGD distribution before and after evolution.