DailyQA: A Benchmark to Evaluate Web Retrieval Augmented LLMs Based on Capturing Real-World Changes

Anonymous ACL submission

Abstract

We propose DailyQA, an automatically updated dynamic dataset that updates questions weekly and contains answers to questions on any given date. DailyQA utilizes daily updates from Wikipedia revision logs to implement a fully automated pipeline of data filtering, query generation synthesis, quality checking, answer extraction, and query classification. The benchmark requires large language models (LLMs) to process and answer questions involving fast-changing factual data and covering multiple domains. We evaluate several open-source and closed-source LLMs using different RAG pipelines with web search augmentation. We compare the ability of different models to process time-sensitive web information and find that rerank of web retrieval results is critical. Our results indicate that LLMs still face significant challenges in handling frequently updated information, suggesting that DailyQA benchmarking provides valuable insights into the direction of progress for LLMs and RAG systems.

1 Introduction

Large language models (LLMs) has demonstrated its wide range of capabilities in the natural language processing (NLP) domain (Devlin, 2018; Brown et al., 2020) and is extending its influence to more and more domains (Radford et al., 2021; Ramesh et al., 2021; Luo et al., 2022; Singhal et al., 2025; Salinas et al., 2020). However, the world is changing fastly, and the static knowledge in the memory of LLMs is usually not updated in a timely manner (Dhingra et al., 2021). A popular approach to this issue is to use retrieval-augmented generation (RAG) (Lewis et al., 2020) techniques to provide the language model with external knowledge, allowing the model to solve problems using in-text learning methods. However, this approach often relies on external retrievers based on keyword or semantic matching (Robertson and Zaragoza, 2009; Karpukhin et al., 2020; Khattab and Zaharia, 2020; Chatterjee et al., 2024; Guo et al., 2024; McDonald et al., 2018). For time-sensitive queries, highly ranked documents may contain misleading information because they do not fulfill the time constraints, thus limiting the capabilities of the RAG system. So we design a time-sensitive query dataset based on realistic changes to measure the model's ability to adapt to rapidly changing information under time constraints.

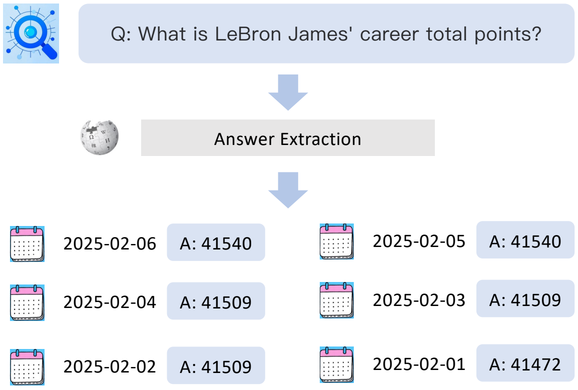


Figure 1: A example for DailyQA. The answer to "LeBron James' career total points" can change every day. For each query in DailyQA, we provide an answer on each day.

Time-sensitive queries have been explored for a long time (Kanhabua and Nørvåg, 2012; Yang et al., 2024b; Gade and Jetcheva, 2024; Mousavi et al., 2024). MRAG (Siyue et al., 2024) added temporal perturbations to the existing datasets TIMEQA (Chen et al., 2021) and SITUATEDQA (Zhang and Choi, 2021) to build datasets with temporal information. UnSeenTimeQA (Uddin et al., 2024), in order to test the model's adherence to temporal information, constructed a virtual dataset. However, they are both static datasets that do not reflect real-time changes in the real world, and thus do not reflect the model's ability to adapt to realistic information. FreshLLMs (Vu

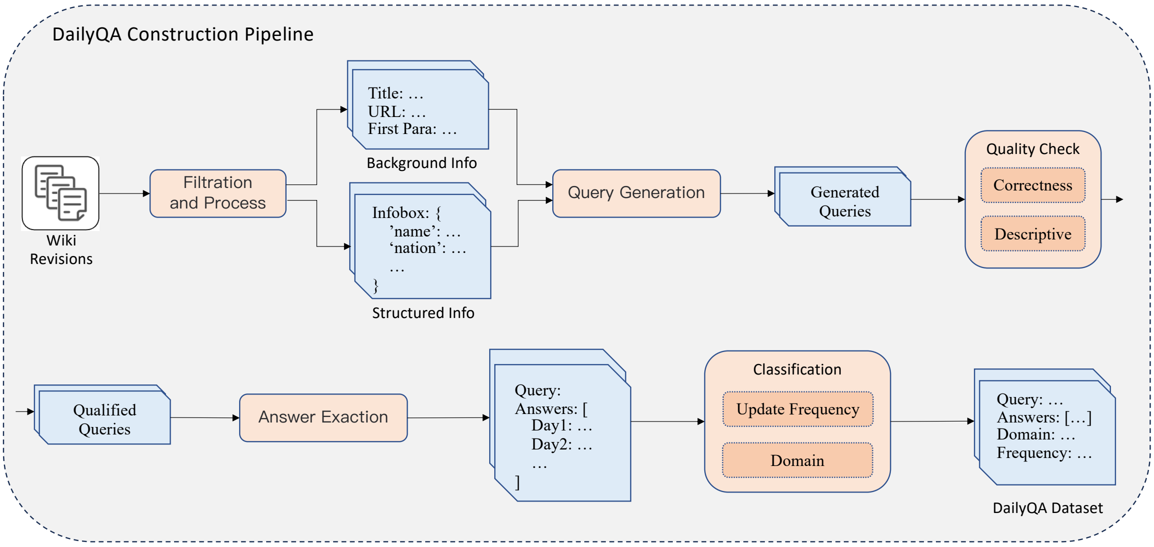


Figure 2: Overview of our DailyQA dataset construction pipeline, which includes filtration and process of the raw data (Wiki revision logs), question generation, quality check, answer extraction, and query classification modules. In the quality check module, we check the correctness and descriptiveness of the queries. In the classification module, we classify queries based on their update frequency and domains

et al., 2023) manually annotated about 600 pieces of data and periodically published updated answers. They can dynamically update the answers, but the queries are static and small in number, so the scope of real-world knowledge involved is limited.

To investigate the capability of LLMs to adapt to complex and changing real-world knowledge, we propose a new benchmark, DailyQA. This work focuses on constructing a daily updated benchmark that contains the latest changes in reality. As shown in Figure 1, each query in DailyQA is provided with an answer on each day. Specifically, we analyze the daily revision records of wiki pages, comparing the page versions before and after the revision and focusing on the changes to the infobox, which tends to contain concise factual information with little redundancy, and then construct the query-answer dataset using the revisions as the golden document. We constructed a stereo measure set by building query data that is updated weekly and corresponding answers that are updated daily. Queries can reflect changes in reality over the span of a week, and paired with the answer to that question on any given day, the time-sensitive nature of LLM can be effectively measured. We designed a fully automated process of data filtering, query synthesis, query quality checking, and answer extraction to ensure the efficient update of the benchmark. In experiments, we measured open-source models such as llama, Qwen, and DeepSeek-R1-Distill-Qwen based on web search augmentation. We also tested them on the DailyQA dataset under a rearrangement that takes into account both temporal and semantic, and found that the performance improved. Our contribution can be summarized as follows:

- We propose DailyQA, a benchmark that responds to changes in reality to measure the adaptability and time sensitivity of LLMs.
- We evaluated several LLMs on DailyQA and proposed an improved rag method for in-context learning. Our experiments show that this task remains challenging for LLMs.
- We analyze further the difficulties in the task of dealing with rapidly changing real-world information, as well as the limitations to LLMs, and then propose promising research issues.

2 Related Works

Time sensitive QA. There has been some work focusing on building time-sensitive benckmarks. MRAG (Siyue et al., 2024) builds a new benchmark on top of the existing dataset TIMEQA (Chen et al., 2021) and SITUATEDQA (Zhang and Choi, 2021) with temporal perturbations. TSQA (Yang et al., 2024b) builds an in-domain Time sensitive dataset for nobel prize. UnSeenTimeQA (Uddin et al., 2024) builds a fictitious, contamination-free benchmark to measure the temporal reasoning ability of the model. However, the domains involved in these works are restricted, and the document corpus they use is static. The scope of knowledge covered by the corpus is too small compared to the information available on the web, which is not a good measure of the adaptability of LLMs in the face of complex and changing information on the Internet. Therefore, we propose DailyQA based on wiki pages covering seven domains such as science and technology, augmented with web retrieval, which is used to measure the adaptability of LLMs in the face of complex web documents.

Realtime QA. FreshLLMs (Vu et al., 2023) manually annotates queries and publishes updated answers weekly, and they create queries of the fast change, slow change, never change, and false promising types. RealTime QA (Kasai et al., 2022) also generates queries using manual annotation, and provides a platform to regularly publish queries and evaluating systems. However, the data size of the queries in the existing work is too small, which leads to a restricted domain and knowledge boundary. For example, FreshLLMs has a fixed query set of 600 queries. RealTime QA updates about 30 queries per week. Such amount of data is too little for reflecting changes in reality as well as for measuring and improving the performance of LLMs. We designed an automatic pipeline to update the benchmark, which can update about 3k queries data per week. And by using our script, readers can easily get answers to queries in the dataset on any given day.

3 DailyQA Benchmark

In this section, we introduce the DailyQA benchmark. In the following subsections, we will introduce the design principles, the build pipeline, and the data structures of DailyQA in turn.

3.1 Benchmark Design Principles

DailyQA focuses on evaluating the ability of large language models to synthesize complex and changing real-world information. For this purpose, we filter and extract valuable information from daily revisions of wikipedia and use it to build a benchenmark that can be automatically updated at low cost.

To reflect the complex and changing reality, we update the set of queries in the benchmark once a week, and for each of these queries, we update its answer every day. In the evaluation phase, we give the query and specify the date, and require that the LLMs, augmented by a web search, have to correctly answer the queries of the corresponding date. This task is challenging and rewarding. Documents obtained through web search may be misleading because they contain information that is too old or too new, which challenges both the reranker and the LLMs. This task is valuable because in real-world scenarios, users might care about factual information about a specific day, for example, 'What is LeBron James' career score as of January 31, 2025?'

3.2 DailyQA Build Pipeline

In this section, we describe the pipeline for building the DailyQA dataset, which includes the following parts: wiki data collection and process, query generation and quality check, and answer extraction.

3.2.1 Wiki Data Collection and Processing

Each time we update the query dataset, we extract all records within a week from the revision records of the wiki and filter them step by step in a rule-based approach. First, we only consider revisions to the main wiki page, i.e. the page indexed by search engines, and ignore revisions to other namespaces. Second, we focus only on revisions in the wiki infoboxes, ignoring changes to other contents. As shown in the Appendix A, this is because wiki infoboxes tend to be well structured and purify factual information with little redundancy compared to the main text. Third, we process the infobox into python's dictionary format, where the content of each block in the infobox corresponds to the value of the dictionary one by one. We further filter based on key and value, that is, we remove keys of setting type such as "color1" and values of filename type such as ending with ".png". For multiple changes to the same page (identified by title), we keep only the last one. We identify changes in terms of key values as the smallest unit, and for multiple changes in a single revision, we keep only one randomly to ensure the diversity of the query set and avoids multiple queries on the same entity.

After the three steps above, we filtered out infobox data that has recently been changed, has good background information (wiki body content), and is well-structured. We store the extracted value (the filtered change), the complete infobox, the title, the url, and the first paragraph of the body text in the wiki page as the extracted data units.

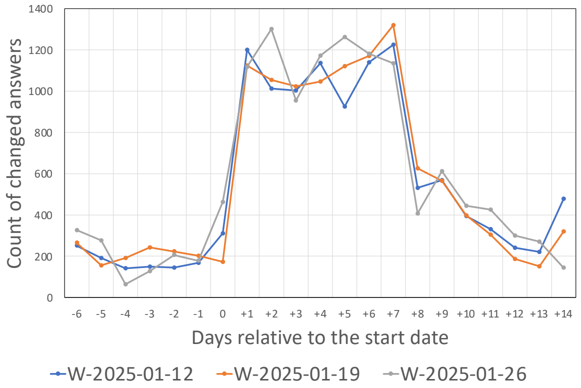


Figure 3: The number of answer changes relative to the previous day. For example, on the line with a start date of 2025/01/12, the "+1" position on the horizontal axis indicates that in the corresponding dataset, the answers for 2025/01/13 was changed by about 1,200 relative to the previous day.

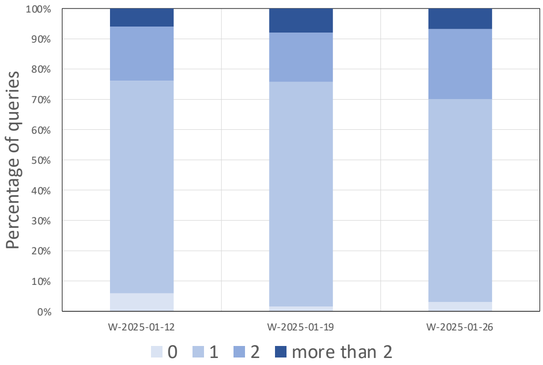


Figure 4: Percentage of queries with different answer change times. For example, as shown in the left bar, in the query dataset for 2025/01/12-2025/01/18, the percentage of queries whose answer change once is about 70%. Note that consistent with Figure 3, we count answer changes over a three-week period that includes the week before and after.

3.2.2 Query Generation and Quality Check

We use a LLM to automatically generate queries. We require the big model to generate a query with the extracted value in a data unit as the answer, and the infobox, the title, and the first paragraph of the body as the background information. The prompt we use is in Appendix B. In this way, we fully describe the context in the query, making the query as precise as possible. At the same time, we map the answer to the value of a block of the infobox, making it easy to extract and update the answer.

In the quality checking stage, we ensure both correctness and descriptiveness of the queries. The correctness of a query means that the query should be able to be answered accurately when sufficient information is provided. The descriptiveness of a query means that the query should be reasonably worded and clearly present the background of the problem, without relying on the background information provided to the LLM.

To check the correctness, we provide the original wiki title, the first paragraph of the body, and the infobox in the data unit as references, and ask the LLM to answer the question based on them. We treat the query as a valid one if the sub match metric between the model answer and the ground truth is 1. To check the descriptiveness, we use DuckDuckGo search api to get the top 10 results and keep only the queries that can successfully retrieve the corresponding wiki pages. In this way, we use external knowledge anchoring to avoid semantic bias.

After the above process, through automatic query generation and quality checking, we automatically obtain a set of correct and descriptive queries that reflect the changes of the reality. On average, we filtered out about 8,000 valid queries from about 11,000 queries per update.

3.2.3 Answer Extraction

According to the above processes, the answer to the question is set to the value of a certain block in the infobox. Therefore, we only need to monitor the corresponding page, the corresponding infobox and the corresponding block every day to get its value to get the answer updated every day. Specifically, we can find the revision history of a page from the wiki logs, and get the correct answer based on the last revision before the requested date. This approach makes it possible to get the answer to a query in the dataset on any day at a very low cost.

3.2.4 Classification

In this section, we introduce the classification of query types for QA datasets. We classify the queries in two perspectives, including their update frequency and domain.

Update frequency. We use the update frequency to mark how often the answer to a query changes. As shown in Figure 3, we statistic the day-by-day variation of answers in the dataset for three updates. Each line in the graph represents an weekly update of queries, for example, "W-2025-01-12" means that this update corresponds to the week starting from 2025-01-12. In the figure, we take the first day of the corresponding week as the start date (day 0 on the horizontal axis). We use the hori-

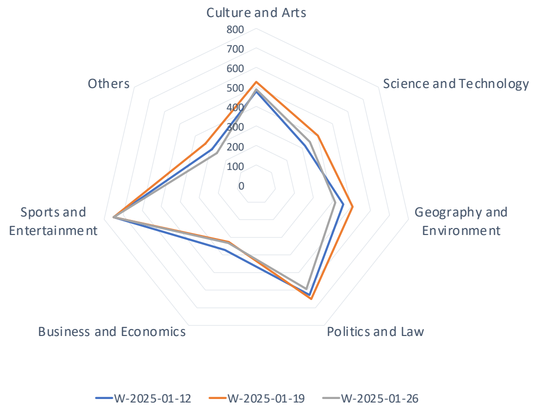


Figure 5: Distribution of the queries in different domains. In the labels, "W-2025-01-12", for example, means a query update corresponds to the week starting from 2025-01-12.

zontal axis to indicate the nth day relative to the start date and use the vertical axis to represent the number of changed answers on that day relative to the previous day. We observe the changes in the answers over a three-week time span and find that the answers change a great deal on day 1-7, while significantly smaller in the week before and after. Since the query is based on the variation on day 1-7, this is as expected. We label queries that do not change from day 8 as "infrequent_update" and the others as "frequent_update".

Above we present the distribution of queries in the dataset at a macro level, and below we will present query-by-query statistics to make it easier for readers to filter and use the parts of their interest. We count the number of answer changes for each query over a three-week period and present it in Figure 3. It can be seen that the number of changes of answers for most of the questions is in the range of 0-1 times, and there are also frequent changes of answers. When answers change infrequently, the difficulty of the queries decreases significantly because web documents tend to include less misleading information. Readers can filter the dataset and use the parts of interest according to their desired difficulty of the task.

Domains. Our query set is based on the comprehensive Wikipedia, so it covers multiple fields. We classify the query set by domain so as to provide convenience for in-domain researches. We use a LLM to classify queries into 7 classes by means of zero-shot, including Science and Technology, Culture and Arts, Geography and Environment, Politics and Law, Business and Economics, Sports and Entertainment, Others. We categorize queries that do not belong to the first six domains as Others. In order to balance the distribution of queries across different domains, we set the maximum number of queries in each domain in each update to 750 and use non-repetitive random sampling method to shrink the oversized query set. The distribution of the data in the different classes is shown in Figure 5. Readers are free to choose the domains of their interest.

3.3 DailyQA Data Structure

After the above pipeline, the data structure of our DailyQA is as follows:

- DailyQA adds a new query dataset every week, which is based on factual information about the latest changes in reality. Our Pipeline automatically crawls the data, generates the query, and check the quality.

- Each query is paired with its update frequency, domain, and golden document (i.e., a Wiki page), which consists of the title, the url, the first paragraph of the body, and a dictionary-formatted infobox.

- Each query's answer is updated daily, which means it has a corresponding answer on any given date. In fact, we provide a script for extracting answers that helps users to obtain answers for a given date easily and cheaply.

4 Experiments

4.1 Baselines

We measured the performance of the RAG system on DailyQA with different web retrieval methods, rag pipelines, and LLMs.

We use **Search w/ Time** and **Search w/o Time** to denote different web search methods. The former means that we add the required date to the query and retrieve the query with the date it over the web, while the latter means that we retrieve the query without the date over the web. By comparing these two methods, we found out the limitations of solving DailyQA directly with the help of search engines.

We compare several types of RAG pipelines. As shown in Table 1, **w/o Search** means that we do not rely on any information retrieved from the web and only rely on the LLM to answer the questions.

		Search w/ Time				Seach w/o Time			
LLM	**Pipeline**	**SM**	**Rouge-L**	**F1**	**Acc**	**SM**	**Rouge-L**	**F1**	**Acc**
Qwen2.5-72B-Instruct	w/o Search	0.120	0.011	0.021	0.139	0.120	0.011	0.021	0.139
	Snippet	0.242	0.159	0.185	0.249	0.263	0.200	0.226	0.286
	Doc	0.356	0.241	0.275	0.364	0.479	0.373	0.410	0.492
	Rerank	**0.392**	**0.308**	**0.338**	**0.416**	**0.502**	**0.413**	**0.446**	**0.513**
	Rerank-T	0.311	0.242	0.268	0.324	0.311	0.250	0.276	0.366
Qwen2.5-7B-Instruct	Rerank	0.350	0.165	0.205	0.366	0.444	0.216	0.264	0.457
Qwen2.5-32B-Instruct		0.351	0.194	0.242	0.364	0.447	0.255	0.312	0.455
DSRD-Qwen-32B	Rerank	0.363	0.101	0.155	0.379	0.433	0.122	0.181	0.452
GPT-4o-mini		0.381	0.209	0.252	0.403	0.484	0.268	0.317	0.498

Table 1: Evaluation on DailyQA with different retrieval methods, RAG pipelines, and LLMs. *Search w/ Time* means web searching queries with dates, and *Search w/o Time* means web searching raw queries. In the RAG pipeline, *w/o Search* means no Web Retrieval Augmentation, *Snippet* means using the web-retrieved snippets as reference, *Doc* means using documents crawled via URLs, *Rerank* means reranking the documents, *Rerank-T* means reranking the documents based on relevance and time. The best results are in **bold** and the second-best are underlined.

Model	**SM1**	**SM2**
Qwen2.5-72B-Instruct	**0.302**	**0.693**
Qwen2.5-32B-Instruct	0.269	0.681
DSRD-Qwen-32B	0.253	0.677
GPT-4o-mini	0.291	0.688

Table 2: SM of the LLMs on frequent_update (SM1) and infrequent_update (SM2) queries. We use the pipeline of *Rerank* and *Search w/o Time* for all the LLMs.

Model	**SM**	**Acc**
Qwen2.5-72B-Instruct	0.466	0.482
Qwen2.5-32B-Instruct	0.445	0.458
DSRD-Qwen-32B	0.434	0.449
GPT-4o-mini	**0.477**	**0.489**
perplexity.ai	0.471	0.485

Table 3: Performance of different LLMs on the DailiyQA dataset in the Science and Technology domain. Except perplexity.ai, we use the pipeline of *Rerank* and *Search w/o Time* for all the LLMs. For perplexity.ai, we provide queries with the specified date and require the service to search for and answer the queries autonomously.

Snippet means that we use the web snippet retrieved by the search engine as the reference, and provide it to the LLM in the order of the web search to help answer the questions. **Doc** means that we obtain the html page based on the URLs returned from the web search, and extract the text of the pages. We then provide them to LLMs in the order of the web search to help answer the questions. **Rerank** denotes that based on the documents of the html pages, we chunk and rerank them, and then provide them to LLMs in the order of reranking. **Rerank-T** means reranking documents based on relevance and time. Based on the "Rerank" pipeline above, we further rerank the chunks with the update time. Specifically, based on the topk document chunks from "rerank", we prioritize the documents whose modification date is before the query date and closer to the query date. By this heuristic approach, we try to provide assistance to LLM in identifying the correct reference documents in the rerank phase.

As shown in Table 1, we evaluated different kinds and sizes of LLMs. Qwen-2.5 (Yang et al., 2024a) series is a set of powerful large language models developed by Qwen that showcase advanced capabilities in natural language understanding and generation. We use Qwen-2.5-72B-Instruct as the base model to evaluate the performance of different rag pipelines. We use the Qwen-2.5 series of models to evaluate the impact of model scale. For closed-source models, we measured the performance of GPT-4o-mini (Achiam et al., 2023). Deepseek-r1 (Guo et al., 2025) is the latest and one of the state-of-the-art LLMs for universal large models. For cost reasons, we measured the performance DeepSeek-R1-Distill-Qwen-32B instead of Deepseek-r1. We use “DSRD-Qwen-32B” to represent the DeepSeek-R1-Distill-Qwen-32B model.

4.2 Metrics

We use the rule metrics and the model evaluation metrics. For the rule metrics, we use subset match (SM), Rouge-L, and F1. The value of SM is 1 if the correct answer is in the prediction and 0 otherwise. For the model evaluation metrics, we require GPT-4o to determine the accuracy (Acc) of the predicted answers. Specifically, we asked the LLM to score

Model	ST	CA	GE	PL	BE	SE	Ot
Qwen2.5-72B-Instruct	0.466	**0.541**	**0.560**	**0.580**	**0.421**	**0.442**	**0.474**
Qwen2.5-32B-Instruct	0.445	0.517	0.532	0.561	0.372	0.413	0.425
DSRD-Qwen-32B	0.434	0.470	0.501	0.497	0.315	0.408	0.348
GPT-4o-mini	**0.477**	0.512	0.538	0.548	0.394	**0.442**	0.447

Table 4: SM of the LLMs on DailyQA in seven domains, including Science and Technology (ST), Culture and Arts (CA), Geography and Environment (GE), Politics and Law (PL), Business and Economics (BE), Sports and Entertainment (SE), Others (Ot).

the similarity of the predicted results to the standard answers, with 5 being completely similar and 1 being completely irrelevant. We computed four and five as correct, i.e., Acc of one, and computed the others as Acc of zero.

4.3 Implementation Details

In the dataset construction phase, we use the pywikibot packet to download and process Wiki logs, and we use Qwen-72B-Instruct to generate queries. We use Qwen-72B-Instruct to answer the queries with golden references for quality check. In the evaluation phase, we use the api of DuckDuckGo as the web search engine, and use Trafilatura to extract the main text in the HTML. We manually specified seven domains and used Qwen-72B-Instruct to identify the domain to which the queries belong. In retrieval enhancement, we uniformly use top 12 snippets, documents or chunks as the reference and use bge-v2-m3 as the reranker. We evaluate on the query update corresponds to the week starting from 2025-01-12, specify the query date as 2025-01-19. We use GPT-4o for the model evaluation.

5 Results

5.1 Main Results

For Qwen2.5-72B-Instruct, **web retrieval is necessary on DailyQA and reranking the raw web-retrieved documents can effectively improve performance**. As shown in Table 1, the results show that the model without web search performs substantially worse than others. This is consistent with our expectations since DailyQA is constructed based on fresh information. Using the original web text is more helpful than using snippets from the search engine, and reranking the raw web-retrieved documents instead of the web retrieval order further improves performance. This suggests that in order to solve this task, we need to keep digging deeper and pay attention to the details of the retrieved content, rather than relying only on summaries. This challenges the information integration capabilities of LLMs and the design of RAG pipelines.

Increasing the scale of the model helps a lot in the metrics of Rouge-L and F1 on DailyQA. The results for different sizes of Qwen2.5 models in the Table 1 show that increasing the model scale leads to a weak improvement in SM and ACC, and a significant improvement in Rouge-L and F1. This means that as the model scale increases, the model tends to be able to answer questions in shorter words, which reflects that the model's answer is more concise with less redundancy. Increasing the scale of LLMs enhances the ability to process time-sensitive realistic documents. It confirms the challenges of DailyQA for LLMs and also illustrates the ability of LLMs to find the required details in complex web references.

Qwen2.5-72B-Instruct works best on DailyQA on all the metrics. We compared several open-source and closed-source models and found that Qwen2.5-72B-Instruct performs best. It outperforms over models on all the metrics. Notably, Qwen2.5-32B-Instruct outperforms DSRD-Qwen-32B on most metrics. DSRD-Qwen-32B, which has been validated to have stronger inference, does not perform as well as the same-sized Qwen2.5-32B-Instruct on this benchmark. This shows that its capability to extract document details is degraded, as well as the possibility of more serious hallucinations. It's suggested that our benchmark is complementary to the LLM evaluation, in the dimension different from the reasoning ability, thus helping to measure LLM's ability more comprehensively.

Our preliminary attempts to integrate time information in the RAG pipeline does not result in a performance improvement. Specifying the date in the web retrieval module and adding time information to the rerank both have a negative effect on the performance. As shown in Table 1, the performance of Search w/o Time is weakly bertter than that of Search w/ Time, and the performance of Rerank is better than that of Rerank-T. This shows that Adding time descriptions directly to the query or rerank the chunks based on time did not result in an improvement. The reason may be that the search engine is not able to accurately understand the intent and process the complex queries so as to return the correct document. This suggests a challenge in calling the search engine more accurately when dealing with time-sensitive real-world problems. Precise retrieval through agentic RAG may be a promising approach in the future.

All models perform better on infrequent update queries than on frequent update queries. As shown in Table 2, We analyze the accuracy of the model on problems with different frequencies of change. The results show thar all models have lower accuracy on the frequent update queries. They are more difficult because documents retrieved from the website tend to include more misleading information, which challenges the ability of LLMs to reason and make temporal judgments. Notice that the gap in model performance is larger on frequent updated quries than on infrequent update quries. This suggests that frequently updated queries are more difficult and that there is more potential for the model to improve on such queries.

DailyQA is a challenge for existing web retrieval augmented LLM services. To measure the difficulty of queries in DailyQA, we measured the it on perplexity.ai and compared it with our methods. As shown in Table 3, perplexity's accuracy on the dataset is comparable to that of our rerank method and there are still about half of the queries that the model cannot answer correctly. This shows that DailyQA benchmark is still a challenge for existing industry solutions.

5.2 Results in Multiple Domains

In order to introduce DailyQA in more detail and to judge the difficulty of the queries in different domains, we measure the accuracy of the models in different domains. The difficulty of the questions varies from one area to another. All the models are relatively more accurate in the domains of Culture and Arts (CA), Geography and Environment (GE), Politics and Law (PL), while they are relatively less accurate within other domains. This is because content in these fields tends to be updated infrequently, while in other fields such as Sports and Entertainment (SE), questions like "What is LeBron James' career total points?" tend to be updated frequently, thus posing a greater challenge.

We find that different series of models have their own areas of specialization. Although Qwen2.5-72B-Instruct has the best overall performance, it does not achieve the best results in all domains. Gpt-4o-mini performs better than Qwen2.5-72B-Instruct in Science and Technology (ST) domain. This implies that due to the different training data and methods, the LLMs may have their own good and bad areas. This provides motivation for building multi-model collaborative agents to solve cross-domain problems.

5.3 Analyse of Challenges

By measuring the performance of the open-source and closed-source LLMs on our benchmark, we can evaluate the ability of these LLMs to process time-sensitive web information. The challenge of this task is mainly twofold.

First, web information is complex and diverse, and it is worth exploring how to fully utilize search engines to obtain the needed information. As shown in Table 1, the modification of adding timestamps by rules may not achieve the expected results, so invoking search engines by issuing queries with the help of LLMs may be a promising direction.

Second, the information in the related documents is time-sensitive. Although the reranked documents have similar semantics with queries, they are likely to contain information that does not meet the time requirement and cause misleading. We have explored methods to rank web pages based on their modification date but it did not result in improvements, possibly because the modification time of a web page is not equivalent to the effective time of the information, and many web pages lack the information of the modification time. Therefore, comprehensively analyzing the retrieved documents and obtaining time information based on content may be a promising direction.

6 Conclusions

We propose DailyQA, a benchmark reflecting changes in reality, to measure LLMs' adaptability and time sensitivity to factual information. We perform the experiments using both open-source and closed-source models and the results show that this task remains a challenge for existing solutions. We further analyze the difficulties in the task of dealing with rapidly changing real-world information, as well as the limitations to LLMs. We expect that by solving the queries in DailyQA, the capabilities of LLMs can be further refined and released.

Limitations

Our benchmark is intended to evaluate the LLMs' ability to process Internet information, and does not focus on the LLMs' logical reasoning ability. Therefore, our dataset contains only one-hop queries and does not include multi-hop queries or false-premising queries.

Due to the limited resources, we did not evaluate the state-of-the-art LLMs such as GPT-o1, DeepSeek-R1, etc. We leave the evaluations on these models for future work.

Affected by the diversity of web page structures, in our implementation, we failed to get the information of the update time for a portion of the web page , so this may degrade the performance of our *Rerank-T* pipeline.

Ethics

This paper constructs a benchmark dataset derived from Wikipedia pages and LLM-generated queries. While Wikipedia content may reflect the personal biases of its contributors, and recently updated pages could occasionally contain unverified information, our methodology mitigates these limitations by exclusively utilizing structured infoboxes as the data source. This approach significantly reduces subjective statements in the referenced Wikipedia texts. Similarly, while LLM outputs may inherit potential biases from training data, our quality check process for queries serves as an inherent quality control mechanism. The dataset is expressly designed for academic benchmarking in NLP research, not for commercial applications. All Wikipedia-derived content remains subject to its original CC BY-SA license terms.

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*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Shubham Chatterjee, Iain Mackie, and Jeff Dalton. 2024. Dreq: Document re-ranking using entity-based query understanding. In *European Conference on Information Retrieval*, pages 210–229. Springer.

Wenhu Chen, Xinyi Wang, and William Yang Wang. 2021. A dataset for answering time-sensitive questions. *arXiv preprint arXiv:2108.06314*.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2021. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273.

Anoushka Gade and Jorjeta Jetcheva. 2024. It's about time: Incorporating temporality in retrieval augmented language models. *arXiv preprint arXiv:2401.13222*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Taian Guo, Taolin Zhang, Haoqian Wu, Hanjun Li, Ruizhi Qiao, and Xing Sun. 2024. Multimodal label relevance ranking via reinforcement learning. In *European Conference on Computer Vision*, pages 391–408. Springer.

Nattiya Kanhabua and Kjetil Nørvåg. 2012. Learning to rank search results for time-sensitive queries. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 2463–2466, New York, NY, USA. Association for Computing Machinery.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Velocity Yu, Dragomir R. Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2022. Realtime qa: What's the answer right now? *ArXiv*, abs/2207.13332.

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*, pages 39–48.

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.

Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. 2022. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in bioinformatics*, 23(6):bbac409.

Ryan McDonald, George Brokos, and Ion Androutsopoulos. 2018. Deep relevance ranking using enhanced document-query interactions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1849–1860, Brussels, Belgium. Association for Computational Linguistics.

Seyed Mahed Mousavi, Simone Alghisi, and Giuseppe Riccardi. 2024. Dyknow: Dynamically verifying time-sensitive factual knowledge in llms. In *Conference on Empirical Methods in Natural Language Processing*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R Pfohl, Heather Cole-Lewis, and 1 others. 2025. Toward expert-level medical question answering with large language models. *Nature Medicine*, pages 1–8.

Zhang Siyue, Yuxiang Xue, Yiming Zhang, Xiaobao Wu, Anh Tuan Luu, and Zhao Chen. 2024. Mrag: A modular retrieval framework for time-sensitive question answering. *ArXiv*, abs/2412.15540.

Md Nayem Uddin, Amir Saeidi, Divij Handa, Agastya Seth, Tran Cao Son, Eduardo Blanco, Steven Corman, and Chitta Baral. 2024. Unseentimeqa: Time-sensitive question-answering beyond llms' memorization. *ArXiv*, abs/2407.03525.

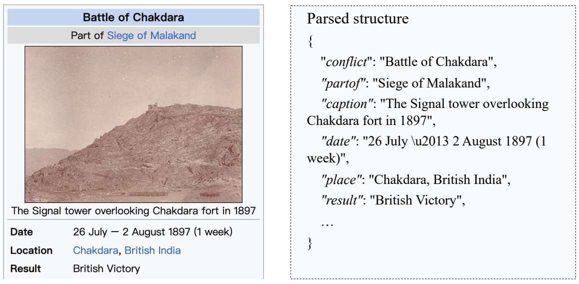
Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and Thang Luong. 2023. Freshllms: Refreshing large language models with search engine augmentation. In *Annual Meeting of the Association for Computational Linguistics*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Wanqi Yang, Yanda Li, Meng Fang, and Ling Chen. 2024b. Enhancing temporal sensitivity and reasoning for time-sensitive question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14495–14508, Miami, Florida, USA. Association for Computational Linguistics.

Michael JQ Zhang and Eunsol Choi. 2021. Situatedqa: Incorporating extra-linguistic contexts into qa. *arXiv preprint arXiv:2109.06157*.

A An Infobox Example



Battle of Chakdara

Part of Siege of Malakand

The Signal tower overlooking Chakdara fort in 1897

Date	26 July — 2 August 1897 (1 week)
Location	Chakdara, British India
Result	British Victory

Parsed structure

```
{
   "conflict": "Battle of Chakdara",
   "partof": "Siege of Malakand",
   "caption": "The Signal tower overlooking
Chakdara fort in 1897",
   "date": "26 July \u2013 2 August 1897 (1
week)",
   "place": "Chakdara, British India",
   "result": "British Victory",
   …
}
```

Figure 6: An example of an infobox from a wikipedia page (left), and its processed data structure (right). The infobox is from the wikipedi *https://en.wikipedia.org/wiki/Battle_of_Chakdara.*

As shown in Figure 6, we introduce An example of an infobox from a wikipedia page, and its processed data structure. We focus only on the infobox structure in the wikipedia page in data processing, and process it into a python dictionarywith the help of the pywikibot tool, which facilitates the information extraction and the understanding of LLMs in the query generation process.

B Prompts

B.1 Prompt for Query Generation

Prompt for Query Generation

Based on the given wiki infobox, ask a question whose answer is the value [{{value}}] in the infobox.
The key of the value in the infobox is [{{name}}].
The infobox is [{{infobox}}].
The infobox is from a wiki page with title [{{title}}] and url [{{url}}].
When asking the question, summarize the background of the question according to the following paragraph and the above infobox to clarify the question.
The background to the question is [{{first_paragraph}}]
The question needs to fulfill the following requirements:
1. when answering the question according to the text [{{infobox}}], the answer is [{{value}}].
2. I will additionally set the background time of the question, so do not include a time restriction (such as on January 1, 2015), in the generated question.
Output the question directly, don't output the answer, don't explain your output, don't output anything else.
The question is:

Figure 7: Prompt for Query Generation

B.2 Prompt for RAG

Prompt for RAG

This is the retrieval augmented generation scenarios where you need to answer questions based on the references.
Note that the information in the references is time-sensitive, and your answer should match the time requirements in the query.
The reference document is as follows: [{{reference}}].
You need to answer the user's question using the original text of the reference without any modification, generalization or summarization.
Answer the questions directly without outputing analysis or additional descriptions.
Question: [{{query}}]
Answer:"

Figure 8: Prompt for RAG

C Examples for Generated Queries

Domains	Queries
SE	What is the total number of Spanish speakers, including those with limited capacity and students learning the language?
CA	How many accolades did the film "Oppenheimer" win?
GE	What is the elevation of Turah, Montana, in feet?
PL	Who was the victim of the child-on-child murder that took place in Walton, Liverpool, England, and how old was he?
BE	What was the production period for the Foton View Kuaiyun, a variant of the Foton View series of light commercial vans?
SE	How many caps has Mario Pašalić made for Atalanta since joining the club in 2020?
Ot	How many children did Edward Fairfax Neild Sr. have?

Figure 9: Examples for generated queries in different domains, including Science and Technology (ST), Culture and Arts (CA), Geography and Environment (GE), Politics and Law (PL), Business and Economics (BE), Sports and Entertainment (SE), Others (Ot).