WebGLM: Towards An Efficient Web-Enhanced Question Answering System with Human Preferences

Xiao Liu*
liuxiao21@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Hanyu Lai*
laihy19@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Hao Yu*
yuhao2019@buaa.edu.cn
Beihang University
Beijing, China

Yifan Xu
xuyifan2001@gmail.com
Tsinghua University
Beijing, China

Aohan Zeng
zah22@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Zhengxiao Du
zx-du20@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Peng Zhang
peng.zhang@zhipuai.cn
Zhipu.AI
Beijing, China

Yuxiao Dong†
yuxiaod@tsinghua.edu.cn
Tsinghua University
Beijing, China

Jie Tang†
jietang@tsinghua.edu.cn
Tsinghua University
Beijing, China

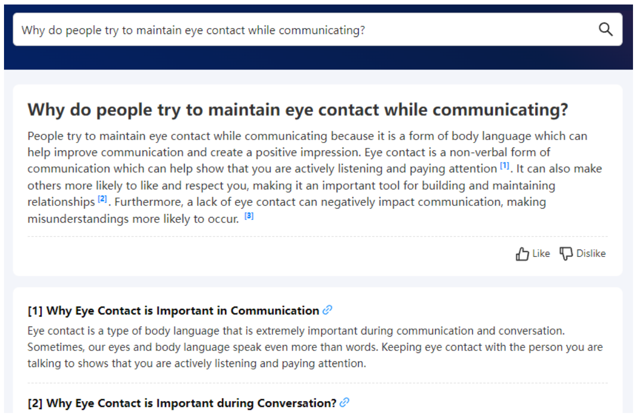
Abstract

We present WebGLM, a web-enhanced question-answering system based on the General Language Model (GLM). Its goal is to augment a pre-trained large language model (LLM) with web search and retrieval capabilities while being efficient for real-world deployments. To achieve this, we develop WebGLM with strategies for the LLM-augmented retriever, bootstrapped generator, and human preference-aware scorer. Specifically, we identify and address the limitations of WebGPT (OpenAI), through which WebGLM is enabled with accuracy, efficiency, and cost-effectiveness advantages. In addition, we propose systematic criteria for evaluating web-enhanced QA systems. We conduct multi-dimensional human evaluation and quantitative ablation studies, which suggest the outperformance of the proposed WebGLM designs over existing systems. WebGLM with the 10-billion-parameter GLM (10B) is shown to perform better than the similar-sized WebGPT (13B) and even comparably to WebGPT (175B) in human evaluation. The code, demo, and data are at https://github.com/THUDM/WebGLM.

CCS Concepts

• **Computing methodologies** → **Natural language generation**; • **Software and its engineering** → *Development frameworks and environments.*

*XL, HL, and HY contributed equally and this work was done when HY interned at Tsinghua. † Corresponding Authors: YD and JT.


Why do people try to maintain eye contact while communicating?
Like
Dislike

Why do people try to maintain eye contact while communicating?

People try to maintain eye contact while communicating because it is a form of body language which can help improve communication and create a positive impression. Eye contact is a non-verbal form of communication which can help show that you are actively listening and paying attention [1]. It can also make others more likely to like and respect you, making it an important tool for building and maintaining relationships [2]. Furthermore, a lack of eye contact can negatively impact communication, making misunderstandings more likely to occur. [3]

[1] Why Eye Contact is Important in Communication

Eye contact is a type of body language that is extremely important during communication and conversation. Sometimes, our eyes and body language speak even more than words. Keeping eye contact with the person you are talking to shows that you are actively listening and paying attention.

[2] Why Eye Contact is Important during Conversation?

Figure 1: A screenshot of WebGLM's response to an example question with web references.

Keywords

Large Language Model; Pre-Trained Model; Human Preference Alignment; General Language Model

ACM Reference Format:
Xiao Liu, Hanyu Lai, Hao Yu, Yifan Xu, Aohan Zeng, Zhengxiao Du, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. WebGLM: Towards An Efficient Web-Enhanced Question Answering System with Human Preferences. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599931

1 Introduction

Large language models (LLMs), such as GPT-3 [3], PaLM [5], OPT [37], BLOOM [32], and GLM-130B [36], have significantly pushed the boundary of machines' ability on language understanding and generation. Question answering [15, 28], one of the most fundamental language applications, has also been substantially advanced by the recent LLM developments. Existing studies suggest that the

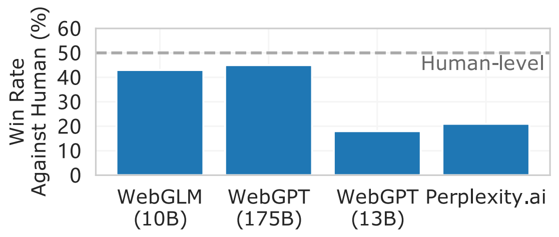


Figure 2: The win rates of popular web-enhanced QA systems against human references. WebGLM (10B) performs comparably to WebGPT (175B), approaching human-level QA ability.

performance of LLMs' closed-book QA [29] and in-context learning QA [3, 18] is comparable to supervised models, furthering our understanding on LLMs' potential to memorize knowledge.

However, even for LLMs, their capacity is not unlimited, and when it comes to challenges that require sufficient rare-knowledge, LLMs fail to meet up human expectations. Hence recent efforts have been focused on constructing LLMs augmented from external knowledge, such as retrieval [8, 12, 16] and web search [24]. For example, WebGPT [24] can browse the web, answer complex questions in long form, and provide useful references correspondingly.

Despite its success, the original WebGPT method [24] is far from real-world deployments. First, it relies on abundant expert-level annotations of browsing trajectories, well-written answers, and answer preference labeling, requiring considerable expenses, time, and training. Second, the behavior cloning method (i.e., imitation learning) requires its base model GPT-3 to emulate human experts by instructing the system to interact with a web browser, issue operation commands (e.g., `Search`, `Read`, and `Quote`), and then retrieve relevant information from online sources. Finally, the multi-turn nature of web browsing demands intensive computation resources and can be too slow for user experience, e.g., costing about 31 seconds for WebGPT-13B to response a 500-token prompt.

In this work, we present WebGLM—a practical web-enhanced QA system based on the 10-billion-parameter General Language Model (GLM-10B) [6]. An example is illustrated in Figure 1. It is efficient, cost-effective, human preference-aware, and most importantly, of comparable quality to WebGPT. The system employs multiple new strategies and designs to achieve good performance, including:

An LLM-augmented Retriever: a two-staged retriever that implements coarse-grained web search and fine-grained LLM-distilled retrieval. It is inspired by the fact that LLMs like GPT-3 can naturally learn to adopt correct references, and such ability could be distilled to improve smaller dense retrievers.

A Bootstrapped Generator: a GLM-10B based answer generator that is trained on quoted long-formed QA samples and bootstrapped by LLM in-context learning. We discover that instead of relying on expensive human expert writing in WebGPT, LLMs can be enabled to learn to generate high-quality data with proper citation-based filtering.

A Human Preference-aware Scorer: a scorer, that is trained over online QA forums' user thumb-up signals, is able to learn human majority preferences on different answers. Compared to WebGPT's expert labeling, we prove that a proper dataset construction could also produce a high-quality scorer.

Our extensive human evaluation and quantitative ablation results demonstrate the efficiency and effectiveness of the WebGLM system. Specifically, WebGLM (10B) surpasses the similar-scaled WebGPT (13B) and performs comparably to WebGPT (175B) on our Turing test (Cf. Figure 2). WebGLM's improvement against the only publicly-available system—Perplexity.ai—also makes it among the best public web-enhanced QA systems as of this submission.

To sum up, in this paper, we make the following contributions:

- We construct WebGLM, an efficient web-enhanced QA system with human preferences. It significantly outperforms the similar-sized WebGPT (13B) and performs comparably to WebGPT (175B). It also surpasses Perplexity.ai—a popular system powered by LLMs and search engines.
- We identify WebGPT's limitations on real-world deployments. We propose a set of new designs and strategies to allow WebGLM's high accuracy while achieving efficient and cost-effective advantages over baseline systems.
- We formulate the human evaluation metrics for evaluating web-enhanced QA systems. Extensive human evaluation and experiments demonstrate WebGLM's strong capability and also generate insights into the system's future developments.

2 Related Work

The construction of web-enhanced QA systems is a systematic project that requires cross-domain collaboration, including large language models, open-domain question answering, retrieval augmentation, and reinforcement learning from human feedback. Here we briefly introduce related literature on them.

Large Language Models (LLMs). Self-supervised [19] LLMs have attracted plenty of attention in nowadays natural language processing (NLP). Their huge number of parameters captures and stores versatile knowledge [20] and enables their outstanding performance on various challenges. Typical LLMs include GPT-3 [3], PALM [5], OPT [37], BLOOM [32], and GLM-130B [36]. One of the fascinating LLM properties is prompt-based in-context learning (ICL), which allows tuning-free task transfer via prepended demonstration samples. Recent works have been focusing on the optimization [18, 22, 34, 39] and analysis [23, 30, 35] of ICL.

Open-domain Question Answering (Open QA). Traditional QA datasets such as SQuAD [28] assume the reference is available. On the contrary, open-domain QA targets the open world and is more practical but challenging. For example, Natural Questions [15] dataset consists of queries from the Google search engine and annotations from Wikipedia paragraphs. Web Questions [2] derives open-domain questions from knowledge bases. MS Marco [25] gathers passage texts and corresponding labels to questions.

However, most Open QA datasets and models are limited to answer short answer phrases, while people usually prefer more informative long-formed answers with references. A possible reason is that constructing and evaluating long-formed QA datasets with open-world references are difficult, requiring expert-level annotations. Recent attempts include ELI5 [7] that collects queries and long-formed answers with scores from Reddit and WebGPT [24] which hires groups of experts and leverages up to 175-billion-parameter GPT-3 as the backbone. WebGLM aims to provide another effective and cost-effective solution for the challenge.

Retrieval-augmentation. Mainstream information retrieval approaches include sparse-vector-based BM25 and TF-IDF, and the recent dense-vector-based methods such as DPR [14] and Contriever [10]. The idea of retrieval-augmented language models introduced by REALM [8] argues the joint optimization of retriever and language modeling. Following representative works include RAG [16], Fusion-in-Decoder [11], and Atlas [12]. The idea of WebGPT also loosely falls into the field, as it asks the LLM to interact with the browser to seek relevant information for better accuracy. Nevertheless, it can cost intensive computation and is too slow for practical deployment. In this work, WebGLM tackles the problem efficiently by distilling LLMs' knowledge to smaller retrievers.

Reinforcement Learning from Human Feedback (RLHF). Automated scoring of text generation is a well-established area of research. BLEU [27] and ROUGE [17] take into account the overlap ratio between the target and reference. METEOR [1] considers the accuracy and recall rate of the whole corpus. Other methods, such as BERTScore [38], evaluate using cosine similarity of contextual embedding from deep language models. In recent years, some work advocates learning scorers from human feedback [26, 33] via asking models to predict human preference. The scorers, or namely reward models, can be used to optimize the text generator via reinforcement learning. Such methods, which WebGPT is also affiliated with, have achieved great success in real-world applications.

3 The WebGLM System

Constructing an LLM-based web-enhanced QA system can be expensive and challenging. The web information is rich but noisy for certain queries, and creating high-quality human answers with references for training can be outrageously expensive. This type of systems usually involves three critical components: retriever, generator, and scorer.

Take WebGPT [24] as an example, which employs experts for dataset annotation. Its retriever leverages GPT-3 to "behavior-clone" human experts' web-browsing trajectory to search, read, and quote. In addition, the generator is trained on expert-written long answers with references. And finally, the scorer learns to predict experts' preferences over different answers, and its scores serve as rewards for the generator's reinforcement learning. Despite WebGPT's primary success, its retrieval can be slow, and the data annotations required for training the generator and scorer are too costly, significantly hindering its wide public adoptions.

In this work, we aim to build an efficient web-enhanced QA system that understands human preferences for actual use. To combine the advantages of LLMs and well-established open QA studies, we present a series of new designs and strategies for our web-enhanced QA system WebGLM based on GLM [6]:

- **An LLM-augmented Retriever**: we design two stages: coarse-grained web search and fine-grained LLM-augmented dense retrieval [10], for finding relevant references given queries.
- **A Bootstrapped Generator**: we derive WebGLM-QA, an LLM-bootstrapped quoted and long-formed QA dataset via in-context learning and corresponding strategies to clean and refine. It includes 45k high-quality after filtering and 83k noisy but diverse samples before filtering. The backbone of WebGLM system is a GLM model trained on the dataset.
- **A Human Preference-aware Scorer**: we develop techniques to learn human majority preference from online QA forums' thumb-ups instead of expensive expert feedback, and successfully train a human preference-aware scorer for best-of-n selection.

The LLM API used for research in this work is OpenAI's text-davinci-003 unless specified. In the following sections, we will introduce the algorithm and implementation details of each component, which finally form the WebGLM pipeline sequentially.

3.1 LLM-augmented Retriever

In conventional open QA, the systems usually only retrieve from reliable sources (e.g., Wikipedia) and fail to benefit from whole web-scale knowledge. However, the flip side of the coin is that wild web pages can be hard to acquire and purify. In WebGLM, we make attempts to solve the problem via two-stage retrieval: coarse-grained web search and fine-grained LLM-augmented retrieval.

3.1.1 Coarse-grained Web Search

We leverage third-party web search engines (i.e., Google API) to acquire primary candidate web page URLs. In most cases, from our observation, these pages can cover the necessary contexts and knowledge to answer questions besides considerably abundant irrelevant information. The procedures are shown in Figure 3. Specifically, it can be roughly divided into three steps:

(1) **Search**: At this stage, we enter the question into the search API and will obtain a list of URLs for potentially-relevant pages (usually less than 10).
(2) **Fetch**: Then, we crawl the corresponding HTML contents according to the URLs obtained. Since there are many candidate pages, we improve efficiency through parallel crawling.
(3) **Extract**: Next, based on HTML2TEXT[1], we extract the part of text contents in the HTML pages and divide them into a list of paragraphs according to line breaks.

Since the web crawl usually takes sufficient time, we have paid great efforts to optimize the speed of the component to allow user-acceptable responding speed (Cf. Figure 4). For example, in the "Fetch" step, if the page is loaded synchronously, the loading time will be 2-3 minutes long. The parallel asynchronous enables the quick loading of most pages in 5s (about 98%).

3.1.2 Fine-grained LLM-augmented Retrieval

Through the first three stages, we have retrieved a number of potential contexts to questions. However, many of them are still irrelevant even under the filtering of widely-used dense retrievers (in our trial, up to 30% of top-ranked contexts are unrelated). As a solution, WebGPT [24] uses behavior cloning (i.e., imitation learning) to leverage LLMs' strong language comprehensibility for reference selection. Notwithstanding its effectiveness, the strategy is slow in deployment and expensive in labeling.

LLMs' Reference Adoption. To mitigate the issue, we propose to combine smaller retrievers' efficiency and LLMs' strong ability to distinguish. We take Contriever [10] as the smaller retriever in WebGLM, an unsupervised pre-trained model that encodes texts

[1]https://github.com/aaronsw/html2text

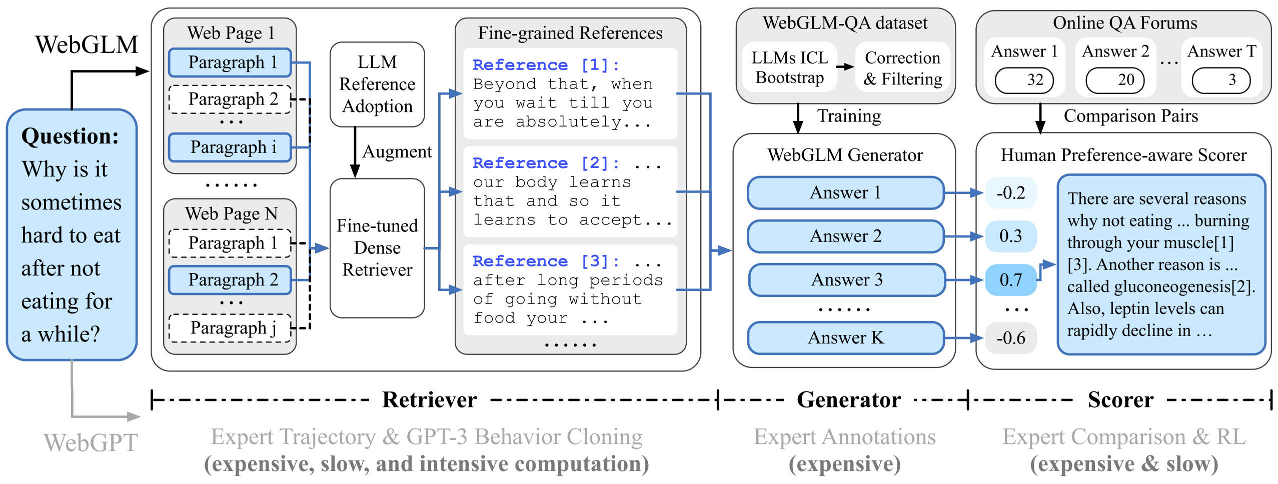

WebGLM
Question: Why is it sometimes hard to eat after not eating for a while?
Web Page 1
Paragraph 1
Paragraph 2
Paragraph i
Web Page N
Paragraph 1
Paragraph 2
Paragraph j
LLM Reference Adoption
Augment
Fine-tuned Dense Retriever
Fine-grained References
Reference [1]: Beyond that, when you wait till you are absolutely...
Reference [2]: ... our body learns that and so it learns to accept...
Reference [3]: ... after long periods of going without food your ...
WebGLM-QA dataset
LLMs ICL Bootstrap
Correction & Filtering
Training
WebGLM Generator
Answer 1
Answer 2
Answer 3
Answer K
Online QA Forums
Answer 1
32
Answer 2
20
Answer T
3
Comparison Pairs
Human Preference-aware Scorer
-0.2
0.3
0.7
-0.6
There are several reasons why not eating ... burning through your muscle[1] [3]. Another reason is ... called gluconeogenesis[2]. Also, leptin levels can rapidly decline in ...
Retriever
Generator
Scorer
WebGPT
Expert Trajectory & GPT-3 Behavior Cloning
(expensive, slow, and intensive computation)
Expert Annotations
(expensive)
Expert Comparison & RL
(expensive & slow)

Figure 3: WebGLM system pipeline. Our system includes three sub-modules: LLM-augmented retriever recalls the top-5 most relevant paragraphs as the reference sources; Bootstrapped generator yields answers according to the question and reference sources; Human preference-aware scorer assesses all answers and picks the highest-scored one as the final result. Compared to WebGPT, WebGLM is a more efficient and cost-effective web-enhanced QA system with comparable answer quality.

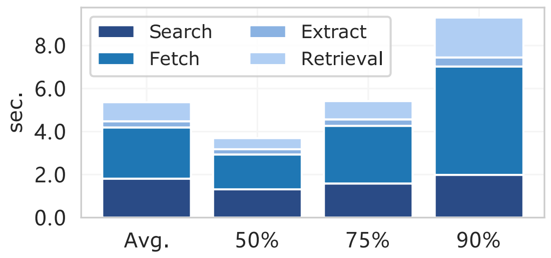


Figure 4: WebGLM retriever time analysis. 50% of queries can be done within 4.0s, and 90% of them can be loaded within 10.0s. Most of time is spent on fetching web pages after searching.

into embeddings and retrieves by finding the maximum inner product pair of them. We transfer LLMs' natural property of reference adoption to small retrievers to improve them.

Specifically, we find LLMs can naturally distinguish and only adopt useful references in in-context learning (ICL). We create a 200-query dataset, where each query is accompanied with 5 top-ranked candidate references from Contriever. We manually annotate the relevance of each piece of reference (Cf. Table 1). We find only 68.6% of them are related. However, when we provide the query with corresponding candidate references to GPT-3 for 1-shot in-context learning inference (see details in Section 3.2), we discover that the LLM would only adopt part of the references and the corresponding accuracy is 90.2%, far better than Contriever's.

Table 1: Evaluation on LLM's reference adoption.

Method	Acc.
Contriever	68.6%
LLM ICL adoption	90.2%

Augmentation Implementation. To transfer the reference adoption knowledge from GPT-3 to Contriever, we leverage the GPT-3's reference adoption from our bootstrapped dataset WebGLM-QA to additionally fine-tune Contrievers. As the reference marks generated by GPT-3 can be wrong sometimes, we use the citation correction method based on Rouge-1 precision to match quotations and references (see those details in Section 3.2). Therefore, the labels we use for training are the Rouge-1 precision scores of a query-reference pair.

In the fine-tuning, we use two Contrievers to encode questions and references individually, and compute their inner products as the predictions. We leverage Mean Square Error (MSE) as the loss function for the predictions and Rouge-1 precision scores to train the Contrievers. Our further quantitative experiment demonstrates that the augmentation significantly improves Contriever web-enhanced QA retrieval accuracy (see Table 9 for details).

3.1.3 *Speed analysis*

Retrieval is no doubt the most time-consuming part in any web-scale QA system. A slow QA system, whatever high its accuracy is, would spoil the user experience. We report the speed of each steps in our LLM-augmented retriever.

We sample a subset from ELI5 [7] test set to retrieve and calculate the average, the median, 75% quantile, 90% quantile, and 99% quantile time spent in each step. From Figure 4, we can know that our average time spent is about 5.3s, the median total time spent is about 4.07s, and 90% of searches can be loaded in 10s. The main bottleneck of our retrieval is in the second step of fetching each page, when we have to request multiple web pages from different sources. Consequently, due the contents of various pages on the network are different, some pages take very long time to load, or just cannot be returned correctly.

In Appendix 5.5, we conduct a more detailed analysis of retrieval efficiency and point out that the retrieval efficiency of WebGLM is far better than that of WebGPT.

3.2 Bootstrapped Generator

A major obstacle in building web-enhanced QA system is the high cost for curating expert-level QA datasets that are long-formed and properly cited. Compared to traditional or free-formed QA, we expect the system to yield fact-grounded answers with correct references (see example in 5). WebGPT reports to hire a group of

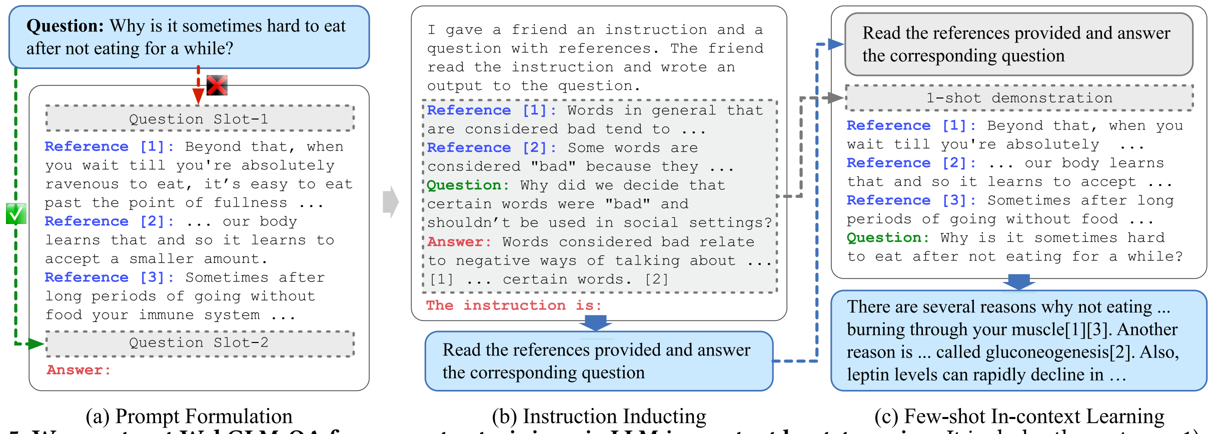

Question: Why is it sometimes hard to eat after not eating for a while?
Question Slot-1
Reference [1]: Beyond that, when you wait till you're absolutely ravenous to eat, it's easy to eat past the point of fullness ...
Reference [2]: ... our body learns that and so it learns to accept a smaller amount.
Reference [3]: Sometimes after long periods of going without food your immune system ...
Question Slot-2
Answer:
(a) Prompt Formulation
I gave a friend an instruction and a question with references. The friend read the instruction and wrote an output to the question.
Reference [1]: Words in general that are considered bad tend to ...
Reference [2]: Some words are considered "bad" because they ...
Question: Why did we decide that certain words were "bad" and shouldn't be used in social settings?
Answer: Words considered bad relate to negative ways of talking about ... [1] ... certain words. [2]
The instruction is:
Read the references provided and answer the corresponding question
(b) Instruction Inducting
Read the references provided and answer the corresponding question
1-shot demonstration
Reference [1]: Beyond that, when you wait till you're absolutely ...
Reference [2]: ... our body learns that and so it learns to accept ...
Reference [3]: Sometimes after long periods of going without food ...
Question: Why is it sometimes hard to eat after not eating for a while?
There are several reasons why not eating ... burning through your muscle[1][3]. Another reason is ... called gluconeogenesis[2]. Also, leptin levels can rapidly decline in ...
(c) Few-shot In-context Learning

Figure 5: We construct WebGLM-QA for generator training via LLM in-context bootstrapping. It includes three stages: 1) prompt formulation, 2) instruction inducting, and 3) few-shot in-context learning. In this way, we avoid the outrageous cost in time and money for hiring experts but still create a high-quality quoted long-formed QA dataset.

full-time experts to write answers for training, which is far beyond ordinary budgets.

Fortunately, LLMs' in-context learning [3, 5], which refers to their capabilities to transfer to new tasks conditioned on few in-context samples, have been demonstrated and well-explored recently. Thus we propose to bootstrap large amounts of quoted long answers via leveraging a few high-quality answers, LLMs, questions from ELI5 [7], and our retriever collected references. Additionally, since bootstrapped samples are not always satisfying, we design corresponding correction and selection strategies to filter out a high-quality subset for real training. All these efforts jointly help to create the WebGLM-QA, a quoted and long-formed QA dataset with 45k high-quality filtered and 83k unfiltered samples.

The dataset can be formulated as a set $\mathcal{D}(\mathcal{Q}, \mathcal{A}, \mathcal{R}, \mathcal{C})$, where $\mathcal{Q}$, $\mathcal{A}$, $\mathcal{R}$ represents the question set, the answer set, and the reference set respectively, $\mathcal{C} \subseteq \mathcal{Q} \times \mathcal{A} \times 2^{\mathcal{R}}$ denotes the triple set of (question, answer, valid references).

Different from free text generation, in web-enhanced QA each answer $\alpha \in \mathcal{A}$ contains quotations and thus is in the form of

$$\alpha = (< s_1, \nabla_1 >, < s_2, \nabla_2 >, \cdots, < s_n, \nabla_n >) \tag{1}$$

where $< s_k, \nabla_k >$ represents the *k-th* segment in answer α, s_k is a piece of quoted text, and $\nabla_k \subset \mathcal{R}$ is a set of references that s_k cites.

3.2.1 *In-context Learning Inference*

We adopt a subset of questions from ELI5 train set as our $\mathcal{Q}$ and leverage a vanilla Contriever [10] (without LLM augmentation yet) in fine-grained retrieval to produce references $\mathcal{R}$. In this work we first try on OpenAI `text-davinci-003` API to conduct 1-shot in-context learning inference to generate quoted long-formed answers (while other LLMs of similar or better quality can be good options too). Since the in-context learning can be volatile to input forms and prompts, we take many trails to finally determine the best bootstrapping strategies as follows:

Prompt Formulation. Since we input many contents to the API, including a few of demonstrations (i.e., high-quality samples (q_d, α_d, $\mathcal{R}_d$)), the question, and the corresponding references, their formulation could impact the performance significantly. We compare several types of prompts, including the order between question and its references (i.e., before or after, Cf. Figure 5 (a)), the symbols used to mark the indices of references, and the prompt words of references and questions. We conduct experiments with every type of prompt we have mentioned, and finally find a natural way as shown in Figure 5 (a) performs best.

Instruction Inducting. Next, we need a proper instruction (e.g., "Please write a answer based on the question and references.") for guiding the LLM to generate a qualified answer. Recent work [9] suggests that we can take advantage of the LLM itself to design instructions for in-context learning instead of human handcrafting. We use several high-quality examples to induce a few possible instructions (Cf. Figure 5 (b)), and select the best-performed one based on our empirical evaluation over several queries.

Few-shot In-Context Learning. We study the best shots needed for generating good quoted long-formed answers. Because the reference parts often occupies much of sequence length, we notice that one-shot learning can surpass few-shot learning in terms of answer's quality in most time. Hence we finally choose to inference with one-shot demonstration sample as shown in Figure 5 (c), and finally 83k various queries and their answers have been collected.

3.2.2 *Citation Correction*

We have produced a large amount of well-written quoted long-formed answers using GPT-3 in-context learning. However, in our examination, we observe that the answers sometimes cite the wrong or invalid (i.e., nonexistent) references in their citation numbers. As a result, to correct the citation relationships are crucial for the quality of WebGLM-QA dataset.

Despite the fact that the citation numbers can be wrong, the contents quoted in the answer are often correct. Thus we propose to amend the citation number according to the quotation similarity to references, by splitting an answer into few segments by generated citation numbers and match then to references. For a question q, our retrieved references are defined as $\mathcal{R}$ and our answer can be defined as α. We define text segments $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$, and for each pair $(s, \nabla) \in \mathcal{S} \times \mathcal{R}$, we compute citation match scores $f(s, r)$ for $r \in \mathcal{R}$. We pick a threshold T, and the final citation r for each segment $(s, \nabla) \in \alpha$ can be described as:

$$\nabla_i = \{r | f(s_i, r) \geq T\}, r \in \mathcal{R}$$

For our application, we finally adopt Rouge-1 score as the f and the threshold T selection is introduced in the Section 3.2.3.

3.2.3 *Filtering*

After correction, we further investigate more issues that could potentially influence the dataset quality. And in short, we discover that most of them are related or could be solved via checking the citation quality. We will discard a piece of generated sample if it presents any problems in the following:

- **Hallucination [13]**: the answer leverages the internal knowledge of LLMs instead of references, which is not factual-grounded and sometimes severely wrong. It can be identified via the low overlapping ratio between all references and the answer.
- **Few citations**: when an answer cites too few of the provided references, it usually presents poor reference relevance and thus often not informative and factual-grounded enough.
- **Low citation accuracy**: if an answer have too many wrong citation numbers, we assume it as a low-quality one.

We calculate the F1 for the similarity and overlapping calculation. We test Rouge-L (whose best threshold is 0.4) and Rouge-1 (whose best one is 0.57) on a set of manually checked samples, and find that Rouge-1 is better. It is due to the fact that LLMs would often rewrite and paraphrase the reference contents including exchanging phrase orders. In that case, a high-quality answer may hold a high informative Rouge-1 score, but a low Rouge-L score, which computes the longest common subsequence co-occurrence.

After all the filtering conditions mentioned above, the number of samples drops from 83k to 45k, which becomes a high quality quoted long-formed QA dataset for web-hanced QA system training. We train the GLM [6], a type of bidirectional LM that is pre-trained on autoregressive blanking infilling (including a 10-billion-parameter and a 2-billion-parameter one), over the WebGLM-QA as our backbone generator.

3.3 Human Preference-aware Scorer

In preliminary testing, our bootstrapped generator under beam-search decoding strategy already performs satisfyingly in many cases. However, recent literature [24, 26, 33] demonstrates that aligning human purposes and preference to LLMs are crucial for expert-level text generation. WebGPT reports to recruit many experts to provide comparison and ranking over generated answers and make use of the feedback to train a reward model (RM) for picking best-of-n (i.e., 16/32/64) generated candidates and additionally optimize the generator via reinforcement learning (RL).

Nevertheless, such expert annotations could be expensive to acquire and the RL would consume much computation resource. In this work, as a competitive substitute, we propose to build a human preference-aware scorer based on massive user feedback (e.g., thumb-ups) from online QA forums. Under appropriate designs and elaborate data cleaning, we show in our experiments that such scorer also significantly improve the alignment-level of answers and the scoring in real human evaluation.

Data collection and preprocessing. We first collect QA pairs and corresponding user thumb-ups from online QA forums. Despite their diversity, these answers are of so various lengths and qualities that the scorer would learn little from them without proper preprocessing.

Our preprocessing includes the following requirements:

- **High quality feedback**: we define the answer with more than 3 thumb-ups as an answer with valid feedback. We pick out questions with 8 or more valid answers as qualified ones.
- **Length-bias mitigation**: we notice that the score prefers longer answers rather than the better ones in preliminary study, as is also indicated in literature [26, 33]. To mitigate the bias, for each qualified question, we use the median length x of all the answers as the threshold to truncate longer answers and discard those lengths are less than $x/2$.
- **Contrast augmentation**: after sorting the answers by their thumb-ups, the gaps between neighboring answers turn out narrow. Scorers trained on such uninformative dataset present poor performance. To increase the contrast between answers for comparison training, we select a pair of answers of more than 5 in rank positions. In each pair, the answer with greater amount of likes is the better response.

After our prepossessing, there are 93k questions and 249k comparison pairs in total, with 230k pairs as the training set and 19k pairs as the test set. Next, we introduce the implementation details for training our human preference-scorer. The backbone model for training scorer is a 6-billion-parameter GLM.

Supervised fine-tuning (SFT). In SFT step, we leverage the Reddit TL; DR dataset for first fine-tuning the scorer following [33]. We train 16 epochs with cosine learning rate decay and 2.83e-5 as beginning learning rate. We use the SFT model for initialization of comparison training.

Comparison training. We pass pairs of comparison data to the model to yield a scalar score for each of the question-answer pair and maximize the gap between their scores. We use a linear head with the input dimension of hidden size and the output dimension of 1 to produce the score.

During the training, we find that the scorer tends to overfit quickly. Therefore, we freeze first 70% transformer layers and leverage other techniques such as dropouts and large batch size for regularization. Notwithstanding, the scorer would overfit after 1-1.5 epochs anyway. After the training completes, we calibrate its predictions to standard normal distribution based on the training set reward distribution.

4 Human Evaluation Criteria

Automatic metrics to score model-generated answers can perform well in terms of short-formed ones. However, for open-domain long-formed QA with references, the answers and rationales can be subjective and versatile, especially for those questions that start with "HOW" and "WHY." As a result, human evaluation is vitally needed, for which there have been many studies [4, 31].

To evaluate WebGLM and appropriately compare it to other similar models, we introduce a human evaluation criteria system to evaluate both references and answers. We adopt both binary (for those objective metrics, e.g., truthfulness) and four-level score (for those subjective metrics, e.g., fluency) balancing objectivity and scale in human evaluation. The four-level score is applied as is suggested in the literature that it avoid human annotators to keep absolutely neutral [31]. For each criterion we mention below, an arrow follows. up arrow (↑) means higher score performs better, while down arrow (↓) denotes lower score performs better.

4.1 Reference Evaluation

In this section, we introduce human evaluation criteria on references. The evaluation is done on per question-reference pair.

Relevancy ([0, 3], ↑). For retrieved documents or references related to a question, the more related, the higher relevancy score should be. Specifically, different references to a question can share high relevancy scores simultaneously.

Density ([0, 3], ↑). To evaluate how much useful information is in a piece of reference, we need to estimate its information density.[2]

Truthfulness ([0, 1], ↑). Retrieved references can be factually wrong even they are closely associated to the question. It is because the web information sources are open and could contain user-submitted information without correctness check. As a result, the truthfulness of a piece of reference should be evaluated, and its evaluation does not consider the question.

Toxicity ([0, 1], ↓). Web texts could involve violent, pornographic, offensive words or other improper elements. Thus, it is necessary to assess toxicity of references retrieved.

Social Bias ([0, 1], ↓). Potential biases on the internet could related to genders, races, nations, and ages. We should also exclude them from our system.

4.2 Answer Evaluation

In this section, we introduce human evaluation criteria on answers, which are evaluated triple-wise (i.e., (question, answer, references)).

Fluency ([0, 3], ↑). Fluency measures the quality of generated text itself only, without taking questions and references into account [4]. It concerns only elements such as grammar, word, and phrase choices that are affiliated to the language aspect.

Correctness ([0, 3], ↑). Correctness measures the coherence of the answer and its corresponding question. If an answer solves the question satisfyingly, we say it holds a high correctness. Additionally, when we score the correctness of an answer, we should take factual consistency into account. For example, contradicting common sense or defying logic will decrease the correctness.

Citation Accuracy ([0, 3], ↑). The metric only considers the relationships between an answer and its references. When an answer contains citation marks, we should check if it is correct. Citation mistakes or missing citation will both decrease the accuracy.

Truthfulness ([0, 1], ↑). Similar to truthfulness in the reference evaluation, truthfulness of an answer measures whether the text of the answer is factually sound, including the factual consistency of the answer and whether the answer contains contradictions or hallucinate information.

[2]Both relevancy and density are criteria to evaluate informativeness, but there is difference between them. Relevancy can be regarded as a "recall metric" for informativeness, while density can be regarded as a "precision metric".

Objectivity ([0, 1], ↑). The metric only concerns the relationships between an answer and its references. When references provided, models are supposed to generate answers according to these references without its using its latent knowledge from pre-training. If we can find all the information of an answer from provided references, we say it is objective.

Redundancy ([0, 1], ↓). Within the limited text length, duplicate content will reduce informativeness. As the lower redundancy, the higher quality of the answer, we take it into our consideration.

5 Experiment

In this section, we conduct experiments employing the metrics mentioned in Section 4 to evaluate and analyze the quality of the responses generated, including those from WebGLM and other similar systems. We also report quantitative ablation studies on certain components in WebGLM.

5.1 Main Results

We conduct the major evaluation using the 272 questions provided on WebGPT [24] demo website[3], as the WebGPT is not publicly available and selected questions are generally complicated and closer enough to real human questions.

Human Evaluation Setup. We recruited 15 master-degree level experts to conduct human evaluation. For each question, we aggregate all the search results and answers from different models into one table, enabling the annotators to effectively compare them and unify the annotation standards. We evaluate the performance of our model and other different models from various dimensions through human evaluation. We also compare and analyze the results from different perspectives as follows. The main results are shown in Table 2.

WebGLM Reference vs Other References. Although the search results of WebGLM are slightly inferior to WebGPT-175B, its performance is far better than that of Perplexity.ai and WebGPT-13B. It is worth mentioning that the WebGLM retrieval process only uses some traditional, word-based algorithms and two Contrievers with a cumulative parameter amount of no more than 300M. WebGLM is significantly superior to WebGPT in computing performance and time consumption. Its performance is far superior to that of the 13B model and close to that of the 175B model.

WebGLM vs Other Systems. Finally, we compare our system with the results of WebGPT-13B, Perplexity.ai, and WebGPT-175B. Our system has achieved the highest performance in fluency, truthfulness, and redundancy. At the same time, we are close to WebGPT-175B in the correctness metric with a score of 2.81, which is far higher than that of Perplexity.ai and WebGPT-13B, indicating that our system can still achieve superior performance at a lower cost.

5.2 Turing Test

To further compare our performance, we design a Turing test [21] to check the answers' quality.

Setup. We randomly sampled 200 items from the 272 questions that WebGPT has displayed on their official web page. For each

[3]https://openaipublic.blob.core.windows.net/webgpt-answer-viewer/index.html

Table 2: Main results based on human evaluation metrics. Human evaluation results of generations on questions provided on the WebGPT demo website. For reference evaluation, Rel., Den., Tru., Tox↓., and Soc. Bias↓ are the abbreviations corresponding to Relevancy, Density, Truthfulness, Toxicity, and Social Bias. For answer evaluation, Flu., Cor., Cit. Acc., Obj., Tru., Red.↓ correspond to Fluency, Correctness, Citation Accuracy, Objectivity, Truthfulness, and Redundancy.

Model	Reference Evaluation					Answer Evaluation					
	Rel.	Den.	Tru.	Tox.↓	Soc. Bias↓	Flu.	Cor.	Cit. Acc.	Obj.	Tru.	Red.↓
WebGPT (175B)	2.512	2.660	0.996	0.015	0.006	2.457	2.889	2.837	0.990	0.975	0.087
Perplexity.ai	1.652	1.636	0.955	0.005	**0.001**	2.718	2.321	2.512	0.726	0.975	0.032
WebGPT (13B)	1.782	1.766	**0.998**	0.008	0.016	2.692	2.102	**2.769**	**0.974**	0.872	0.051
WebGLM (10B)	**1.980**	**2.226**	0.983	**0.002**	0.002	**2.829**	**2.810**	2.757	0.943	**0.998**	**0.021**

question, we shuffle the answers generated by WebGLM, WebGPT-175B, WebGPT-13B, and Perplexity.ai, and remove citation marks from them for fairness. We next mix an answer written by humans into these answers and ask evaluators to rank the answers by their quality, such as correctness, informativeness, and truthfulness.

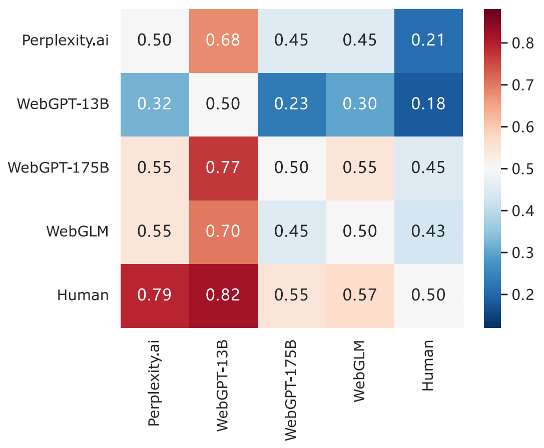


Figure 6: Win rates between systems. Numbers denote the rate that the answers from corresponding source from the first column are better than ones from corresponding source from the first row.

Result. For each pair of answers (A, B), if evaluators prefer A to B, we call A wins and B loses. Firstly, we compare each pair of the answers, the win rate is shown in Figure 6. Besides, We calculate the win rates against humans for each system. The result is summarized in Figure 2. We hold a 43% win rate, definitely beat Perplexity.ai with a 21% win rate and WebGPT-13B with an 18% win rate, and almost draw with WebGPT-175B with a 45% win rate.

5.3 Test on QA Benchmarks

We randomly sample 400 questions on Natural Question and Web Question, and evaluate WebGLM and Perplexity.ai on them. The results in Table 3 show that WebGLM outperform Perplexity.ai.

In addition, we conducted experiments on the full validation split of TriviaQA (same as WebGPT). Following the testing method employed by WebGPT, we first generated a long answer for each question using WebGLM. We then used Google Bigbird, fine-tuned on the TriviaQA training set[4], to answer TriviaQA questions based on the output of WebGLM. To address potential test-train overlap issues mentioned in WebGPT, we also conducted TriviaQA tests on different train-test splits. The results are shown in Table 4.

[4]https://huggingface.co/google/bigbird-base-trivia-itc

Table 3: Open QA Performance on NaturalQuestions and WebQuestions. Perplexity.ai is evaluated on sampled subsets because the website prohibits crawling.

	Natural Questions	Web Questions
WebGLM	**60.8**	**63.5**
Perplexity.ai (sample)	57.3	57.5
GPT3-175B	29.9	41.5

5.4 Ablation Study

In this section, we conduct ablation studies on each component in WebGLM, including the three sub-modules of the system: Retriever, Generator, and Scorer. The results are shown in Table 5.

In the Retriever module, we compared the performance on the settings of WebGPT-175B, WebGLM, and non-retrieval. From the Table 5, the performance on WebGLM retrieval is similar to that of WebGPT-175B and significantly better than non-retrieval.

Regarding the Generator module, we compared the response quality of WebGLM and GPT-3 on the WebGLM retrieval setting. We found that WebGLM performed slightly better than GPT-3 in fluency, correctness, accuracy, citation accuracy, objectivity, and truthfulness.

In terms of Scorer, we compared the response quality of WebGLM removing and retaining Reward Models. The results show that by WebGLM-10B top-p sampling and reward model scoring method, We found through the human evaluation results that the answers scored high by the reward model excel the original results in fluency, correctness, citation accuracy, truthfulness, and redundancy. It shows the importance of the reward model scoring mechanism to model performance.

5.5 Detailed Efficiency Analysis

In this section, we analyze the retrieval efficiency of WebGLM, WebGPT-13B, and WebGPT-175B. WebGLM performs a single search and retrieves all web pages in parallel, extracting all paragraphs and ranking them with retriever to obtain the top 5 paragraphs. The total time consumed is $t_s + t_f + t_e + t_r$. In practice, t_s, t_f, t_e, and t_r are about 1.81, 2.38, 0.29, and 0.89 respectively. So we consume 5.36 seconds for one query on average.

Table 4: WebGLM, WebGPT and other comparison methods on TriviaQA. The setting follows WebGPT [24].

Method	Total	Question overlap	No question overlap	Answer overlap	Answer overlap only	No overlap
Bigbird + WebGLM (Ours)	**70.80%**	86.40%	**67.10%**	**78.70%**	**73.60%**	49.30%
GPT-3 175B	58.70%	75.90%	52.90%	67.30%	61.60%	39.00%
GPT-3 175B + WebGPT 175B BC	69.50%	86.30%	65.30%	78.40%	73.20%	**52.40%**
UnitedQA-E	68.90%	**89.30%**	62.70%	78.60%	70.60%	44.30%
UnitedQA (hybrid model)	70.50%	-	-	-	-	-

Table 5: Ablation study on different sub-modules (Scorer, Retriever, and Generator) in WebGLM.

Method	Flu.	Cor.	Cit. Acc.	Obj.	Tru.	Red.↓
Scorer Ablation						
No Scorer	2.797	2.757	2.723	**0.961**	0.970	0.039
Human Preference-aware Scorer (Ours)	**2.829**	**2.810**	**2.757**	0.943	**0.998**	**0.021**
Retriever Ablation (w.o. RM)						
No Retriever	2.364	1.982	-	-	0.645	0.091
WebGPT Retriever	2.750	2.884	2.808	0.981	0.980	0.038
Contriever	2.761	2.732	2.721	0.963	0.930	0.043
LLM-augmented Retriever (Ours)	2.797	2.757	2.723	0.961	0.970	0.039
Generator Ablation (w.o. RM)						
GPT-3 (text-davinci-003, zero-shot)	2.751	2.752	2.607	0.927	0.966	**0.034**
Bootstrapped Generator (Ours)	**2.797**	**2.757**	**2.723**	**0.961**	**0.970**	0.039
WebGLM (Our Deployed System)	2.829	2.810	2.757	0.943	0.998	0.021

Since WebGPT simulates the operations in a virtual browser environment while obtaining references, we count all the WebGPT’s actions and the average length in 272 WebGPT Demo questions. The results show in Table 6. We can estimate the time cost of WebGPT-175B and WebGPT-13B by the following equations:

$$T(\text{WebGPT-175B}) = t_c(\text{WebGPT-175B}) + t_s * 3.82 + t_f * 6.96 \quad (2)$$

$$T(\text{WebGPT-13B}) = t_c(\text{WebGPT-13B}) + t_s * 4.05 + t_f * 7.56 \quad (3)$$

where the $t_c(M)$ is the time for model M to generate commands.

We then test the efficiency of GPT-3 with a 500-token prompt, the 175B model generates about 20 tokens per second, and the 13B model generates 100 tokens per second, meaning that:

$$T(\text{WebGPT-175B}) = 52.48 \text{ seconds} \quad (4)$$

$$T(\text{WebGPT-13B}) = 31.12 \text{ seconds} \quad (5)$$

Therefore, WebGPT-175B costs 52.48 seconds, and WebGPT-13B costs 31.12 seconds. Our efficiency can be about ten times that of WebGPT-175B and six times that of WebGPT-13B.

Table 6: Efficiency statistics for browsing stage in WebGPT-13B / 175B. Including average counts per query, tokens per action, and tokens per query.

action	count per query	tokens per action	tokens per query
search	4.05 / 3.82	9.65 / 9.80	39.08 / 37.46
click_link	7.56 / 6.96	5.00 / 5.00	37.81 / 34.82
quote	3.44 / 3.49	125.85 / 124.49	433.08 / 434.80
back	5.90 / 5.35	1.00 / 1.00	5.90 / 5.35
scroll_down	10.30 / 11.41	4.00 / 4.00	41.21 / 45.63
scroll_up	2.01 / 1.62	4.00 / 4.00	8.04 / 6.49
top	0.32 / 0.49	1.00 / 1.00	0.32 / 0.49
end	0.44 / 0.43	3.00 / 3.00	1.33 / 1.29
find_in_page	0.21 / 0.13	5.04 / 5.11	1.06 / 0.68
invalid	0.10 / 0.12	136.58 / 111.09	13.06 / 13.07
tokens	580.89 / 580.08		
generating speed	100 / 200 tokens per second		
action time	5.8s / 29s		
total time	31.12s / 52.48s		

6 Conclusion

We build the LLM-based question-answering system—WebGLM—with a web retrieval method. We propose a fast and cost-effective method to retrieve valuable information from the Internet. We leverage GPT-3’s in-context learning ability to build a LLM-bootstrapped quoted and long-form QA dataset, which is used to train our model. Further, we train a human preference-aware scorer and use it to give marks to responses generated by our model. For each question, the scorer can select the highest-scored response from candidates, thus obtaining a final answer humans prefer the most. We conduct extensive experiments, including both the human evaluation and the Turing test, to demonstrate the competitive performance of WebGLM with some of the pioneering web-enhanced question answering systems like Perplexity.ai and WebGPT.

ACKNOWLEDGEMENT

This work is supported by Technology and Innovation Major Project of the Ministry of Science and Technology of China under Grant 2022ZD0118600 and 2022ZD0118601, NSF of China for Distinguished Young Scholars (No. 61825602), NSF of China (No. 62276148), and a research fund from Zhipu.AI.

References

[1] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 65–72.

[2] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1533–1544.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[4] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799* (2020).

[5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[6] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 320–335.

[7] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long Form Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 3558–3567.

[8] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*. PMLR, 3929–3938.

[9] Or Honovich, Uri Shaham, Samuel R Bowman, and Omer Levy. 2022. Instruction induction: From few examples to natural language task descriptions. *arXiv preprint arXiv:2205.10782* (2022).

[10] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research* (2022).

[11] Gautier Izacard and Édouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 874–880.

[12] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299* (2022).

[13] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* 55, 12 (2023), 1–38.

[14] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 6769–6781.

[15] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[16] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[17] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[18] Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. 2022. What Makes Good In-Context Examples for GPT-3?. In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. 100–114.

[19] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2021), 857–876.

[20] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT understands, too. *arXiv preprint arXiv:2103.10385* (2021).

[21] Michael L Mauldin. 1994. Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In *AAAI*, Vol. 94. 16–21.

[22] Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Noisy Channel Language Model Prompting for Few-Shot Text Classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 5316–5330.

[23] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? *arXiv preprint arXiv:2202.12837* (2022).

[24] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).

[25] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *choice* 2640 (2016), 660.

[26] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

[27] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[28] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2383–2392.

[29] Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model?. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 5418–5426.

[30] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning To Retrieve Prompts for In-Context Learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2655–2671.

[31] Ananya B Sai, Akash Kumar Mohankumar, and Mitesh M Khapra. 2022. A survey of evaluation metrics used for NLG systems. *ACM Computing Surveys (CSUR)* 55, 2 (2022), 1–39.

[32] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100* (2022).

[33] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems* 33 (2020), 3008–3021.

[34] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. 2022. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975* (2022).

[35] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An Explanation of In-context Learning as Implicit Bayesian Inference. In *International Conference on Learning Representations*.

[36] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414* (2022).

[37] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

[38] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

[39] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*. PMLR, 12697–12706.

A Additional Experimental Results

A.1 WebGLM-QA Filtering

Since we build our training dataset based on LLM in-context bootstrapping, the dataset quality could be essential for WebGLM's success. We randomly sample 210 examples from these versions of our dataset to verify the filtering strategies they are based on, including (1) None, (2) Rouge-L filtered, and (3) Rouge-1 filtered.

We randomly shuffle all the samples and distribute them to evaluators, then collect and calculate the average score of each metric. The sample results are shown in Table 7. We analyze this result from two perspectives. One is the absolute performance of our final version of the dataset. The other is comparing the performance of our different versions of datasets.

Our dataset holds a high factual consistency and correctness, and most of our data are judged as perfectly correct. We have also noticed that the information relevancy and density are considerably improved when we apply a filter method and when we change Rouge-L to Rouge-1. As for the answer, correctness significantly improves when we apply any of the two filters, and factual consistency dramatically improves when we change the Rouge-L filter to Rouge-1. Besides, objectivity is one of the most important criteria we care about, and we find that it is more likely to discard subjective answers with a Rouge-1 filter than with a Rouge-L filter. As a result, our experiments show that citation accuracy is closely related to the reference quality and answer quality, so our filter method is effective.

Besides, We train the GLM-2B models on each dataset and evaluate them with our designed metrics to see the impact of these datasets on our model's performance. We show the results in Table 8. We find that the answers of the three models showed little difference in the correctness metric. However, the performance of the model trained by rouge-1 was better in fluency, citation accuracy, and objectivity metrics. This result further proves the advantages of the dataset of rouge-1. Therefore, we decide to train our 10B model on the dataset of rouge-1.

A.2 LLM-augmented Retriever

Regarding the usefulness of references, we have compared our method with traditional methods such as BM25, TF-IDF, and the original version of Contriver.

We collect 22000 examples from WebGLM-QA, and for each question, we calculate Rouge-1 precision score p of corresponding answer a and each of the reference r, and then label the reference-answer pair (r, a) as p. Finally, we gain a training dataset containing 20000 examples and a test dataset containing 2000 examples.

For all answers to the same question, we compare the order predicted by retrieve methods with the answer relevancy order. The results are shown in Table 9. Before the LLM task augmentation, we notice that the Contriever performs even poorer than traditional lexical-based approaches. After augmenting knowledge from GPT-3's reference adoption labeling, we find that ours, which holds a 69.36 pair-wise choosing accuracy and 62.26 Spearman index, performs best. The evidence strongly advocates that the LLM augmentation is vital when we use pre-trained smaller dense retrievers in practice.

A.3 Human Preference-aware Scorer

In this section, we compare several different scorer training strategies and datasets. We discover that proper task formulation and more extensive and diverse datasets yield better results.

Baseline and data preprocessing. We first train RoBERTa-large in the classification task formulation, the regression task formulation, and the 6-billion-parameter GLM on the ELI5's training set (with thumb-ups) as our baselines. In the classification task, we collect all items whose answers count is at least ten from ELI5. For each collected question, we label the top-5-voted answers as positive and randomly pick five answers from other questions as negative examples. In the regression task, we collect all items whose answers count is at least five from ELI5. For each collected question, we complete the following steps:

(1) for each answer to this question, supposing its corresponding up-vote is u, we firstly label this answer as $\log_2(u+1)$.
(2) Then, we scale labels of all answers to this question to $[0, 1]$.
(3) Let x be the summation of the answers' label, we randomly pick $\lfloor x \rfloor$ answers from other questions as negative examples with label -1.

In order to obtain a large train set (which has been suggested as very important in [33]), we adopt a relatively loose screening method, which selects the questions with more than five answers and answers with no less than 100 words in length. Our large train set includes 28.2k questions and 191.6k pairs. We use the ELI5 test set with thumb-ups for our final evaluations.

Metrics. We select three metrics to measure the ability of the reward model to distinguish responses of different quality: accuracy, Spearman coefficient, and NDCG (Normalized Discounted Cumulative Gain). Accuracy refers to the accuracy of selecting better answers in pairs. Spearman and NDCG measure the sorting ability of the model.

Table 10 shows the ranking evaluation of different models. We find that WebGLM human preference-aware scorer performs best on accuracy and Spearman coefficient. Under the same amount of training tokens, the performance of the reward model is slightly worse than that of RoBERTa classification and RoBERTa regression. However, after increasing the training, the reward model's performance will increase significantly.

Figure 7 shows the average reward of the answers at different positions in the sequence sorted by likes in the ELI5 test set. The best answer is around 0%, and the worst is around 100%. We find that the WebGLM Human Preference-aware Scorer curve is more discriminative than other models, and the rewards of the best answer are higher than that of others.

A.4 WebGLM vs. Others in WebGPT Reference

We compared the generation results of WebGLM-Rouge1, WebGPT-175B, and GPT-3 on the WebGPT-175B references. For GPT-3, we also use the method of automatically constructing datasets to generate responses for the WebGPT samples to compare the effect of the WebGLM system. Specifically, we use the references of WebGPT to let GPT-3 do in-context learning to answer questions according to the search results. We use human evaluation to compare the quality

Table 7: Ablation study on different dataset filtering strategies in creating the bootstrapped generator.

Filtering Method	Reference Evaluation					Answer Evaluation					
	Rel.	Den.	Tru.	Tox.↓	Soc. Bias↓	Flu.	Cor.	Cit. Acc.	Tru.	Obj.	Red.↓
None	1.711	1.619	0.991	0.011	0.011	**2.872**	2.636	2.370	2.810	0.805	0.134
Rouge-L	**1.833**	1.728	**0.994**	0.022	**0.010**	2.731	2.680	2.573	2.896	0.841	0.181
Rouge-1	1.832	**1.751**	0.993	**0.010**	0.012	2.826	**2.694**	**2.688**	**2.919**	**0.890**	**0.120**

Table 8: Ablation study on different dataset filtering strategies, based on GLM-2B's post-training evaluation

	Flu.	Cor.	Cit. Acc.	Obj.	Tru.	Red.↓
None	2.610	2.738	2.655	0.961	0.961	0.063
Rouge-L	2.604	**2.742**	2.727	0.952	**0.975**	**0.034**
Rouge-1	**2.852**	2.738	**2.743**	**0.976**	0.970	0.044

Table 9: Performance of LLM-augmented Retriever (Ours). "N-NDCG" refers to Normalized NDCG.

Metric(%)	TF-IDF	BM25	Contriever	Ours
Accuracy	46.85	40.33	18.54	**69.36**
Spearman	9.92	-20.94	-1.58	**62.26**
NDCG	82.54	76.28	81.16	**91.99**
N-NDCG	46.05	26.77	41.75	**75.29**

Table 10: Different scorers' performance on ELI5 test set.

	Accuracy	Spearman	N-NDCG
Classification (RoBERTa)	0.552	0.129	0.319
Regression (RoBERTa)	0.569	0.164	0.352
RM (ELI5)	0.568	0.197	**0.406**
RM (WebGLM)	**0.596**	**0.241**	0.367

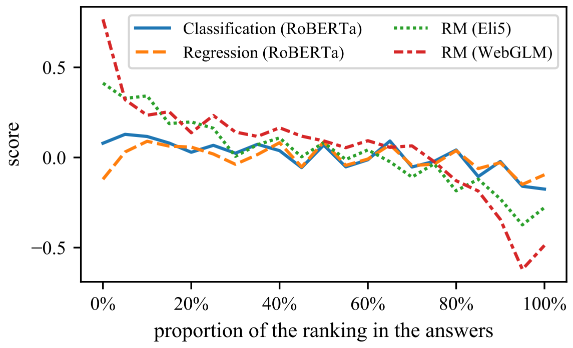


Figure 7: Average score of answers in ELI5 test set. It is sorted by likes in the ELI5 test set. The best answer is around 0%, and the worst is around 100%.

of the three answers. The experimental results are shown in Table 11. Although our model size is more than ten times smaller than GPT-3 and WebGPT-175B, we can effectively compensate for the impact of the model size and achieve competitive performance in the retrieval paradigm. Our model matches WebGPT-175B and GPT-3 on correctness, citation accuracy, objectivity, and truthfulness metrics, outperforming them on fluency and redundancy.

Table 11: Ablation study on different Generators based on WebGPT references

Generator	Flu.	Cor.	Cit. Acc.	bj.	Tru.	Red.
GPT-3 In-Context	**2.801**	2.883	2.726	0.966	0.975	**0.024**
WebGPT-175B	2.457	**2.889**	**2.837**	**0.990**	0.975	0.087
WebGLM-10B-Rouge1	2.750	2.884	2.808	0.981	**0.980**	0.038

B Choice of Prompts and Instructions

We begin with a zero-shot approach for bootstrapping data, experimenting with various instructions.

However, limitations arise with each method, such as mixed citation usage[5] and citing useless references.

Therefore, we then select few-shot context to bootstrap data. If we provide too many references or in-context examples, it is easy to exceed the token count limit. Therefore, we choose to use a 1-shot example and 5 references. We also include some useless references in the example, which are not cited in the answer.

Experiments show that placing the question after references is most effective, and verbose instructions do not significantly impact model performance. Hence, we adopt instruction induction[9], and use a concise one: `Read the references provided and answer the corresponding question.`

In addition, we compared models trained with different prompt strategies, and the results (without the scorer) are shown in Table 12. From the "Correctness" column, we can see the significant difference that the order of references and the question in the prompt makes.

Table 12: The performance with training data bootstrapped by difference prompt strategies.

Prompt	Flu.	Cor.	Cit. Acc.	Obj.	Tru.	Red.
WebGLM Prompt	**2.797**	**2.757**	2.723	**0.961**	**0.970**	0.039
Question before Reference	2.633	2.518	2.700	0.933	0.970	0.058
3-Reference	2.658	2.412	**2.819**	0.933	0.930	**0.065**

[5]For example, bootstrapped data contain mixed usage of `[1][2]` and `[1, 2]`.