More Bang for your Context: Virtual Documents for Question Answering over Long Documents

Yosi Mass, Boaz Carmeli, Asaf Yehudai, Assaf Toledo, Nathaniel Mills
IBM Research AI
{yosimass,boazc}@il.ibm.com,
{Asaf.Yehudai,assaf.toledo}@ibm.com,
wnm3@us.ibm.com

Abstract

We deal with the problem of Question Answering (QA) over a long document, which poses a challenge for modern Large Language Models (LLMs). Although LLMs can handle increasingly longer context windows, they struggle to effectively utilize the long content. To address this issue, we introduce the concept of a virtual document (VDoc). A VDoc is created by selecting chunks from the original document that are most likely to contain the information needed to answer the user's question, while ensuring they fit within the LLM's context window. We hypothesize that providing a short and focused VDoc to the LLM is more effective than filling the entire context window with less relevant information. Our experiments confirm this hypothesis and demonstrate that using VDocs improves results on the QA task.

1 Introduction

Large language models (LLMs) such as OpenAI's ChatGPT[1] or Anthropic's Claude[2], have demonstrated exceptional performance across a range of natural language processing (NLP) tasks and are capable of answering questions on various topics. Recent advancements have extended their context window to encompass tens of thousands or even millions of tokens (Gemini, 2024). This enables them to process long documents, or even books, and answer questions over them.

Despite this impressive progress, recent studies have shown that LLMs still struggle to utilize long contexts efficiently (Li et al., 2024; Levy et al., 2024). LLMs' performance often deteriorates when confronted with a long document, hindering their ability to accurately answer questions on such documents.

To address this challenge, we propose a method called Virtual Document (VDoc). VDoc takes a long document, a question, and a specified context length, and generates a shorter version of the document, containing the most relevant parts for the given length. VDoc operates in three steps: 1) **Document segmentation**: break down the long document into smaller segments, 2) **Segments ranking**: rank the segments based on their relevance to the question, and 3) **VDoc reconstruction**: reassemble the segments into a coherent virtual document that contains the most relevant information to the question, and still fits the given context length. The overall VDoc process is illustrated in Figure 1.

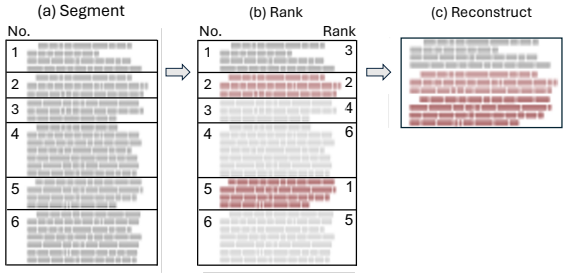

Figure 1: VDoc architecture. First, a long document is segmented (a), then each segment is ranked by its relevance to the question (b), and finally a virtual document is reconstructed with the top-ranked segments (c).

We analyze experimentally, different approaches for each step. Our results show that VDoc enhances question-answering performance on various datasets, including ZeroScrolls (Shaham et al., 2023), a benchmark designed for tasks requiring a long context.

The paper's main contribution is the thorough analysis and experiments of the above three steps for QA over a long document. While the suggested solution is close to RAG (Retrieval Augmented Generation), it has some unique properties. For example, the ordering of the segments based on their sequence in the original document, that does not exist in RAG. Our code is available on git[3]

[1] https://chat.openai.com/
[2] https://www.anthropic.com/news/claude-2-1
[3] https://github.com/IBM/vdoc

Findings of the Association for Computational Linguistics: EACL 2024, pages 12936–12942

2 Related Work

Many previous works focus on designing LLMs that can support long context. Some works modify the transformer attention mechanism (Beltagy et al., 2020; Zaheer et al., 2021; Dai et al., 2019; Ratner et al., 2023) or make it more efficient (Dao et al., 2022; Dao, 2023; Liu et al., 2023a; Gemini, 2024). Others have focused on modifying the positional embeddings (Press et al., 2022; Chen et al., 2023). Another line of research attempts to replace transformers, which have quadratic sequence length complexity, with new architectural designs like convolution and linear RNNs, e.g., RWKV (Peng et al., 2023), S4 (Gu et al., 2022), Hyena (Poli et al., 2023), or Mamba (Gu and Dao, 2023).

While those works were able to extend the context length of LLMs, subsequent research has found that long-context transformers exhibit a recency bias and do not effectively utilize long-range context (Qin et al., 2023; Liu et al., 2023b; Yehudai and Bendel, 2024; Li et al., 2024). Specifically, these models have been found to struggle with ignoring irrelevant information (Levy et al., 2024). Moreover, benchmarks for long context reveal that there is still significant room for improvement compared to human performance, and for some tasks, models even struggle to surpass the naive baseline (Shaham et al., 2023).

3 Method

We deal with the problem of using an LLM to answer questions over a long document. Our method, termed VDoc, creates a shorter version of the document, which is then used as input to the LLM, instead of the entire document, to generate an answer. Creating the VDoc is done in three steps. i) Document segmentation into manageable segments, ii) Segments ranking by their potential to contain the answer to the user question, and iii) VDoc reconstruction. The detailed steps are described below.

Document segmentation. We try two methods. i) sliding-window on sentence boundaries using nltk[4] ii) context-aware (using HTML structure when available). We define a *segment-size* parameter that controls the max segment length.

Segments ranking. We experiment with three families of language models (LM). The first is encoder-only models, where we rank segments by the similarity of their representation to the representation of the user question. An example of such LM is MiniLM[5].

The second, is encoder-decoder or decoder-only models, where we rank segments by the inverse perplexity of the user question, given each of the segments. Specifically, we give each segment as input to the model with the instruction: "Generate a conversation between a user and an agent based on the given content:". We then run a forward loop and compute the Cross-Entropy loss for the question. The segments are ranked by $r = (-1) \times loss()$ (higher is better). An example is flan-t5[6].

The third, is a sparse representation, Elser[7] which expands and indexes documents into ElasticSearch[8] index, with semantic terms that were learned to co-occur frequently with terms in each document.

We note that among the three segment ranking methods, the first (comparing dense representations) and the last (Sparse representation) can be pre-computed on the given set of documents. The second method on the other hand (using perplexity), requires running a forward loop of the model for each segment and question, and thus can not be pre-computed, because it depends on the question. Thus the perplexity method can be quite slow for very long documents.

VDoc reconstruction. We take the top ranked segments, that still fit into the given window size and evaluate two methods to concatenate them: i) by the original order in the document (*Doc order*) and ii) by relevancy to the question (*Rank order*).

4 Experiments

4.1 Setup of the Experiments

We evaluated the VDoc performance in two scenarios: *direct* and *end-to-end*. For the *direct* scenario we assume a benchmark of triplets {(Q, D, E)} where Q is a question, D is a long document and E is an evidence i.e., a span in D that contains the answer. We define the *VDoc task* as follows: Given a context-window size, a question Q, and a long document D, create a virtual document that

[4] https://www.nltk.org/

[5] https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2

[6] https://huggingface.co/google/flan-t5-large

[7] https://www.elastic.co/guide/en/machine-learning/current/ml-nlp-elser.html

[8] https://www.elastic.co/

fits within the context-window size and includes E. We measure the success of the VDoc to contain the evidence (we counted 90% containment as a success, to tolerate for segmentation issues).

For the *end-to-end* scenario, we assume a benchmark of triplets {(Q, D, R)} where Q is a question, D is a long document that contains the answer, and R is a gold response. We then assess LLM's response against the gold, R, when prompting using various VDoc strategies. We evaluate results using F1 (the harmonic mean of Recall and Precision), similar to the Scrolls (Shaham et al., 2023) paper.

We experiment with two LLMs, each with 8K window size, thus we limit the VDoc to sizes of up to 7200 tokens to allow space for the prompt, the question and generated answer. In all experiments, we used segment-size of 512 tokens.

4.2 Datasets

For the *direct* experiment, we utilize two datasets containing long documents, questions and evidence. The first is GoogleNQ (Kwiatkowski et al., 2019). This dataset contains questions from Wikipedia. Each question is labeled with the Wikipedia page that contains the answer, and a span in the document that contains the evidence. The second dataset is taken from Scrolls (Shaham et al., 2023)[9]. Scrolls contains several datasets with questions over long documents. We used Qasper which is question answering over research papers. We added an evidence for each question, from the original Qasper dataset[10].

For the *end-to-end* experiment we utilized the Qasper dataset described above and the NarrativeQA dataset from Scrolls. NarrativeQA contains question-answer pairs about entire books and movie scripts. Both datasets contain (Q, D, R) triplets.

We picked a random subset of 500 examples from the validation sets of Qasper and NarrativeQA. Likewise, for GoogleNQ, we picked 500 examples where the long answer (evidence) is a section.

Note that all our experiments were done on the validation sets of Qasper and NarrativeQA from Scrolls. We used the validation set and not the test set, since ground truths is available only on the validation set. We selected a random subset of 500 examples, inspired by the size of the test sets in ZeroScrolls. We tried the Qasper experiments on the full validation set and got very similar results, thus we performed the experiments on the 500 data points.

[9] https://www.scrolls-benchmark.com/tasks

[10] https://huggingface.co/datasets/allenai/qasper

Dataset	Avg # tokens	Max # tokens
GoogleNQ	12,337	273,087
Qasper	5,484	22,052
NarrativeQA	88,073	516,045

Table 1: Datasets used for evaluation. Number of tokens (#) is calculated by the flan-t5-small tokenizer.

The datasets and their statistics are described in Table 1. The numbers are for the random subset of 500 examples. Note that in all our experiments, we treat each document separately, thus we report statistics only for the subset that was used in the experiments.

4.3 Optimal Window Size

To confirm that a smaller window yields better generation results, we conducted a controlled experiment, using the evidence as a pivot. We expanded the content up to a specified window size, measured in tokens, while maintaining full sentences using the nltk tool.

We experiment with expanding the content on both sides of the evidence, as well as before and after it. All three methods yielded similar results. Figure 2 reports results when expanding the window in both sides of the evidence. We conducted the experiments using two generative LLM: flan-t5-xxl and llama-3-8b-instruct. As observed, both models achieve the highest F1 score when provided only with the evidence, and performance degrades as data is added.

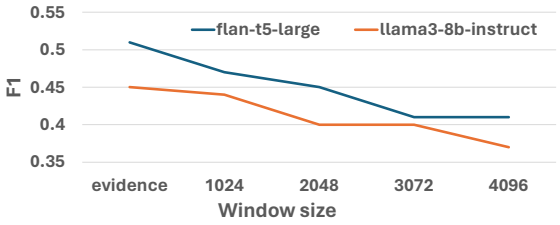

Figure 2: Comparing performances (F1) of response generation on Qasper when adding non-relevant information around the evidence.

4.4 Direct VDoc Evaluation

We compared three segments ranking (perplexity (*flan-t5-large*), *minilm-l12*, and *Elser*) and a base-

VDoc ranker	Qasper-sliding-window			GoogleNQ-sliding-window			GoogleNQ-semantics		
	2400	4800	7200	2400	4800	7200	2400	4800	7200
prefix	43.12	64.13	81.94	84.60	91.17	94.03	84.60	91.17	94.03
flan-t5-large	69.91	89.86	88.89	69.42	81.77	84.70	72.10	84.90	89.55
minilm-l12	50.51	76.81	79.17	57.37	75.21	77.61	65.40	80.34	85.82
Elser	71.46	89.86	**91.67**	76.56	77.78	**96.27**	91.96	97.44	**98.13**
# required VDocs	487	276	72	448	351	268	448	351	268

Table 2: Direct VDoc evaluation. Comparing success (%) of VDoc rankers to capture the evidence in the VDoc. For Qasper we use sliding-window only. For GoogleNQ we use both sliding-window and semantic segmentation.

DB	VDoc ranker	flan-t5-xxl			llama-3-8b-instruct		
		2400	4800	7200	2400	4800	7200
Qasper	prefix	0.25	0.32	0.33	0.25	0.33	0.34
	flan-t5-large	**0.37** (0.35)	0.36 (0.35)	0.35 (0.35)	0.35 (0.34)	0.35 (0.34)	0.35 (0.33)
	minilm-l12	0.28 (0.28)	0.34 (0.33)	0.35 (0.35)	0.31 (0.28)	0.33 (0.33)	0.33 (0.32)
	Elser	0.35 (0.34)	0.36 (0.34)	0.35 (0.35)	**0.37 (0.37)**	0.35 (0.35)	0.34 (0.33)
Narrative	prefix	0.15	0.15	0.15	0.14	0.16	0.17
	flan-t5-large	0.19 (0.18)	0.19 (**0.20**)	0.19 (0.19)	0.19 (0.20)	0.21 (0.21)	0.23 (0.22)
	minilm-l12	0.15 (0.14)	0.16 (0.16)	0.17 (0.17)	0.16 (0.16)	0.20 (0.18)	0.20 (0.20)
	Elser	0.17 (0.17)	0.19 (0.18)	0.19 (0.18)	0.18 (0.19)	0.22 (0.21)	**0.24** (0.22)

Table 3: End-to-end evaluation. Comparing F1 of the generation by flan-t5-xxl and llama-3-8b-instruct on Qasper and NarrativeQA. Each cell contains values for doc order and (rank order).

line *prefix* (that truncates to the window-size), for varying window-size.

To use Elser as a ranker, We pre-index all segments of all documents into an ES index. We add a document-id as a field to all segments of each document. To rank the segments of a given document, we query the index with the given question, and use the document-id as a filter, thus only segments of the given document are scored by Elser.

Results are summarized in Table 2. The last row (# required VDocs), measures the number of cases that the input document was larger than the window size and thus required VDoc activation. For example, when using window-size of 2400, for Qasper, 487 out of the 500 documents were larger than 2400 tokens, thus required VDoc. The rest were smaller than 2400 tokens, thus we could send the full document as is. Similarly, for window-size 4800, only 276 Qasper documents were larger than 4800 tokens, thus required VDoc activation. We can see that the number of required VDocs decreases as the window size increases, and thus the success to capture the evidence in the VDoc increases.

Segments ranking. We observe that on Qasper, all VDoc rankers outperform the naive method of taking the prefix, where Elser and perplexity (with flan-t5-large) achieve the best performance, with Elser performing slightly better. On GoogleNQ, we see that with a sliding-window segmentation, the naive prefix is better than the three VDoc rankers (except for Elser on window-size=7200). This implies that the evidence in GoogleNQ, in most cases, is at the beginning of the document.

Document segmentation. To evaluate a more sophisticated segmentation, we utilized the HTML format of the Wikipedia pages used in GoogleNQ[11]. We cleansed the Wikipedia HTML to remove noise (e.g., headers, footers, sidebars, advertisements) and converted the HTML to semantic passages, identified by the document title and headings preceding the passage. When moving to semantic segmentation, we can see that Elser outperforms the prefix. For example, even on the smaller window size of 2400, it succeded to capture the gold evidence in 91.96% of the cases, compared to 72.10% and 84.60% with perplexity (flan-t5-large) and prefix respectively.

4.5 End-to-End Evaluation

We compared two LLM models with 8K window size: flan-t5-xxl and llama-3-8b-instruct. We use the prompts from ZeroScrolls. For llama3, we

[11] Qasper comes as free text, so we could not apply semantic segmentation on it

modified the prompts a bit. The detailed prompts and an example are given in Appendix A.

Table 3 shows the results on Qasper and NarrativeQA, for varying context-window size (2400, 4800, 7200) and the different segment rankers.

As seen, all VDoc strategies outperform the naive prefix method both on NarrativeQa and on Qasper. For example on NarativeQA, with context-window of 7200, llama-3-8b-instruct achieved an improvement of 41% (0.24 with Elser, compared to 0.17 with prefix). On Qasper we see an improvement of 48%, on context-window 2400, with llama-3-8b-instruct (0.37 with Elser compared to 0.25 with prefix).

Furthermore, we observe a correlation with the direct VDoc evaluation (Table 2). In both experiments, the best results are achieved for Elser and perplexity (flan-t5-large) rankers. A closer examination of the two datasets reveals that on Qasper, those two rankers remain quite stable across different window sizes. On NarrativeQA, we can see an improvement for all segment rankers, when increasing the window size, and all outperform the naive prefix by a large margin.

The behavior of the two datasets supports our hypothesis that it is enough to give the LLM a smaller window and still get comparable results to those with a larger window. On Qasper, the 4800 window is a sweet spot, namely a good balance between capturing the evidence, and yet not adding too much irrelevant information to the LLM. This is because most of Qasper documents fit into the 7200 window. On NarrativeQA, the sweet spot is beyond 7200 since the documents are much larger.

VDoc reconstruction. As shown in Table 3, arranging the top-selected segments in their original order (*Doc order*) generally results in a better F1 score compared to ordering them by relevancy to the question (*Rank order*).

ZeroScrolls evaluation. We submitted flan-t5-xxl on the best VDoc configuration of Elser with window size 4800 to the official ZeroScrolls leader board. With VDoc, our flan-t5-xxl achieved F1 scores of 50.7, 30.6 on Qasper and NarrativeQA respectively, compared to F1 of 48.3 and 19.3 when used without the VDoc. This is an improvement of 58% on NarrativeQA which has very large documents.

4.6 The Effect on Inference Times

In this section, we analyze the contribution of the VDoc on inference times. We used the Llama-3-1-70b-instruct model, which supports an input length of 128K tokens. We run it on the NarrativeQA 500 examples, used in all our experiments.

Figure 3 shows the running times and the F1 as a function of the context-window size. We used the llama-3-1-70b-instruct on IBM watsonx.ai[12] for inference. The times are for generation only. The F1 values are reported for VDoc that uses the best segment-ranking method (see Table 3 above), namely Elser. We can see that the best F1 is achieved with VDoc for context window size of 7200 tokens, and it starts degrading as we increase the context window size. Still we can see that running times for the 500 examples, increase from 28 min (3.36 sec/example) to 125 min with context-window of 64K (15 sec/example).

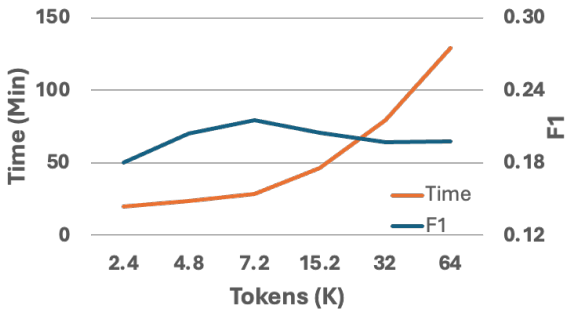

Figure 3: Inference times and F1 of response generation, on NarrativeQA, as a function of context-window size.

5 Conclusions and Future Work

We described a method called VDoc, enabling Question Answering using LLM over long documents. The method is based on creating a virtual document that contains relevant information to the user query, and adheres to the LLM's context-window limitations. Our method does not require any modifications or fine-tuning a LLM.

We compared different methods and showed experimentally, that it is better to aggressively shorten the document, even if the LLM can handle a larger context-window size, both in terms of quality of the results, and in terms of better run times.

[12] https://www.ibm.com/products/watsonx-ai

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness.

Gemini. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces.

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context llms struggle with long in-context learning.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023a. Ring attention with blockwise transformers for near-infinite context.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023b. Lost in the middle: How language models use long contexts.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Bolun Wang, Johan S. Wind, Stanislaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. Rwkv: Reinventing rnns for the transformer era.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation.

Guanghui Qin, Yukun Feng, and Benjamin Van Durme. 2023. The nlp task effectiveness of long-range transformers.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. Parallel context windows for large language models.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. Zeroscrolls: A zero-shot benchmark for long text understanding.

Asaf Yehudai and Elron Bendel. 2024. When llms are unfit use fastfit: Fast and effective text classification with many classes.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2021. Big bird: Transformers for longer sequences.

A Appendix - Detailed Prompts and Examples

In all the end-to-end experiments, we used the ZeroScrolls prompts. For llama-3-8b-instruct, we found however that repeating the last instruction after the query, helps improve its results. Without that addition, llama3 sometimes tends to return very long answers, which hinders precision.

Tables 4 and 5 list the prompts used by llama3 for the Qasper and NarrativeQA respectively. Table 6 illustrates the above phenomena. We can see that without the suffix, llama3 tends to give very long answers. The F1 for the first answer is 0.67 (with recall=1 and precision=0.5), while the values for the second answer are 0.03, 1, 0.02 for F1, recall and precision, respectively.

You are given a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article,write "unanswerable". If the question is a yes/no question, answer "yes", "no", or "unanswerable".

Article:
{input}

Question:
{question} **Answer the question as concisely as you can, using a single phrase or sentence if possible**.

Answer:

Table 4: llama-3 prompt for Qasper. Note the added suffix after the question

You are given a story, which can be either a novel or a movie script, and a question. Answer the question as concisely as you can, using a single phrase if possible.

Story:
{input}

Question:
{question} **Answer the question as concisely as you can, using a single phrase or sentence if possible**.

Answer:

Table 5: llama-3 prompt for NarrativeQA. Note the added suffix after the question

Question: What kind of injustices do the stories point out?

Gold answer: Social injustices.

Answer 1: Social and economic injustices

Answer 2: The stories point out the injustices of social class, wealth, and power. In "The Adventure of the Abbey Grange", the stepfather's abuse of power and wealth leads to the mistreatment of his stepchildren. In "The Adventure of the Black Swan", the lady's freedom is threatened by the five miles of country between her home and Winchester, highlighting the limitations of women's freedom in rural areas. In "The Adventure of the Empty House", Sherlock Holmes criticizes the lack of justice for the poor and ignorant, who are often victims of hidden wickedness and cruelty. The stories also highlight the class differences between the wealthy and the poor, with the wealthy often having more power and influence. Overall, the stories point out the injustices of a society where wealth, power, and social class can lead to exploitation and mistreatment of others.

Table 6: An example question. **Answer 1** is with the added suffix, while **Answer 2** is without.