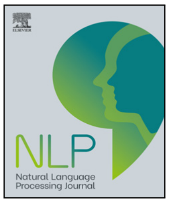
Contents lists available at ScienceDirect

Natural Language Processing Journal

journal homepage: www.elsevier.com/locate/nlp



LeanContext: Cost-efficient domain-specific question answering using LLMs

Md Adnan Arefeen[a,b,*], Biplob Debnath[b], Srimat Chakradhar[b]

[a] *University of Missouri-Kansas City, Kansas City, MO, USA*
[b] *NEC Laboratories America, Princeton, NJ, USA*

ARTICLE INFO

Keywords:
Domain-specific question answering
Cost-efficient LLMs
Large language models
Reinforcement learning
Text summarization

ABSTRACT

Question-answering (QA) is a significant application of Large Language Models (LLMs), shaping chatbot capabilities across healthcare, education, and customer service. However, widespread LLM integration presents a challenge for small businesses due to the high expenses of LLM API usage. Costs rise rapidly when domain-specific data (context) is used alongside queries for accurate domain-specific LLM responses. Extracting context from domain-specific data is implemented by a Retrieval Augmented Generation (RAG) approach. One option is to summarize the RAG context by using LLMs and reduce the context. However, this can also filter out useful information that is necessary to answer some domain-specific queries. In this paper, we shift from human-oriented summarizers to AI model-friendly summaries. Our approach, LeanContext, efficiently extracts k key sentences from the context that are closely aligned with the query. The choice of k is neither static nor random; we introduce a reinforcement learning technique that dynamically determines k based on the query and context. The rest of the less important sentences are either reduced using a free open-source text reduction method or eliminated. We evaluate LeanContext against several recent query-aware and query-unaware context reduction approaches on prominent datasets (arxiv papers and BBC news articles, NarrativeQA). Despite cost reductions of 37.29% to 67.81%, LeanContext's ROUGE-1 score decreases only by 1.41% to 2.65% compared to a baseline that retains the entire context (no summarization). LeanContext stands out for its ability to provide precise responses, outperforming competitors by leveraging open-source summarization techniques. Human evaluations of the responses further confirm and validate this superiority. Additionally, if open-source pre-trained LLM-based summarizers are used to reduce context (into human consumable summaries), LeanContext can further modify the reduced context to enhance the accuracy (ROUGE-1 score) by 13.22% to 24.61%.

1. Introduction

In recent times, large language models (LLMs) have seen extensive utilization (Espejel et al., 2023), especially since the introduction of LLM APIs for customer-oriented applications on a large scale (Liu et al., 2023). These applications include chatbots (like GPT-4), language translation (Jiao et al., 2023), text summarization (Luo et al., 2023; Yang et al., 2023; Zhang et al., 2023a; Bhuyan et al., 2023), and question-answering (QA) tasks (Tan et al., 2023), personalized robot assistance (Wu et al., 2023). While the zero-shot performance of the LLM model is nearly on par with fine-tuned models for specific tasks, it has limitations. One significant limitation is its inability to answer queries about recent events on which it has not been trained. This lack of exposure to up-to-date information can lead to inaccurate responses, particularly for domain-specific information processing, where the LLMs may not grasp new terminology or jargon. To build an effective domain-specific question-answering system, it becomes essential to educate the LLMs about the specific domains, enabling them to adapt and understand new information accurately.

LLMs can learn domain-specific information in two ways, (a) via *fine-tuning* the model weights for the specific domain, (b) via *prompting* means users can share the contents with the LLMs as input context. Fine-tuning these large models containing billions of parameters is expensive and considered impractical if there is a rapid change of context over time (Schlag et al., 2023) e.g. a domain-specific QA system where the documents shared by users are very recent and from different domains. A more practical way is to select the latter approach i.e. the prompt-based solution, where relevant contents from user documents are added to the query to answer based on the context. Motivated by this, we focus on prompt-based solutions for document-based QA systems.

One of the challenges for long document processing using a prompt-based solution is the input prompt length being limited to a maximum length defined by the LLM API. The token limit of GPT-3.5 and GPT-4 vary from 4,096 to 32,768 max tokens limit proportional to the usage cost. Therefore, LLMs will fail to answer the query if the prompt length is larger than the max token limit due to the larger context length in

* Corresponding author.
E-mail addresses: adnanarefeen@mail.umkc.edu (M.A. Arefeen), biplob@nec-labs.com (B. Debnath), chak@nec-labs.com (S. Chakradhar).

https://doi.org/10.1016/j.nlp.2024.100065
Received 8 November 2023; Received in revised form 31 January 2024; Accepted 15 March 2024

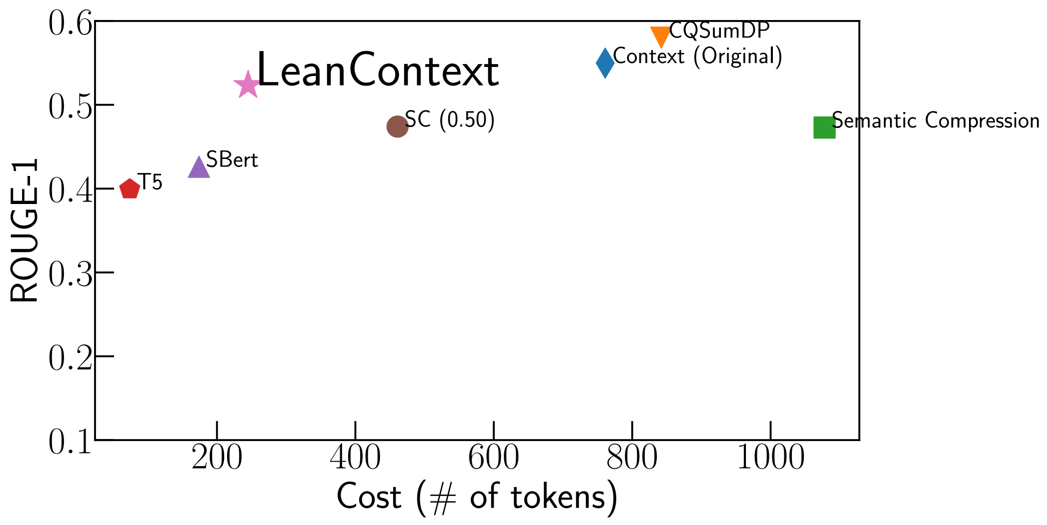


Fig. 1. Compared to the original context LeanContext only drops in ~2% ROUGE-1 score with ~ 68% savings on BBCNews dataset (Li, 2023).

the prompt. One suitable way to get rid of this problem is via document *chunking* (Chase, 2022). In this case, initially, the user documents are segmented into chunks. Only the relevant chunks of fixed size are retrieved as context based on the query. This is known as retrieval augmented generation (RAG) of context (Lewis et al., 2020).

The cost of context-based querying using LLM APIs via prompting is linked to the number of input tokens (contributing to prompt cost) and the number of output tokens (contributing to generation cost). According to a recent study (GPT-3 Cost, 2023), with 15,000 visitors having 24 requests per month, the cost of using GPT-3 (Davinci model) is $14,400 per month (assuming prompt tokens = 1800, output tokens = 80) which is challenging for a small business to operate. For GPT-4, the cost is even higher than this amount. In this paper, our focus is to reduce this cost. To mitigate the LLM API usage cost, the number of tokens in the context should be reduced as the cost is proportional to the length of the context. A low-cost option to reduce the context is to summarize the context using free open-source summarizer models. However, for domain-specific QA applications, the pre-trained open-source summarizers do not contribute to good accuracy. On the contrary, using a pay-per-use model like ChatGPT further increases the query processing cost instead of reducing it as the additional cost is added at the time of text reduction.

To this end, we propose a domain-specific query-answering system LeanContext, where users ask queries based on a document. To answer a query, LeanContext first forms a context from the document based on the query by retrieving relevant chunks. Next, it identifies the top-k sentences related to the query from the context. Subsequently, LeanContext reduces a part of the rest of the sentences in fragments by an open-source text reduction method if the sentences lie in the top-k sentence region set. The rest of the less important sentences are eliminated. Following this, it forms a new context by stitching top-k sentences and reduced fragments in the order of their appearance order in the original context. Finally, it invokes an LLM (like ChatGPT) to answer the query using that new context. Determining the value of k when selecting the top-k sentences can be challenging because the specific value of k varies depending on the query and relevant context. Therefore, instead of using a fixed k, LeanContext employs a reinforcement learning (RL)-based technique to dynamically determine k based on the query and context. It is important to note that LeanContext's objective differs from the summarization task aimed at generating coherent summaries for human readers. In LeanContext, the reduced context is intended for consumption by a question-answering model like ChatGPT. Fig. 1 shows the scenario of LeanContext, reducing the context size with accuracy close to the original context and outperforming other open-source models.

In summary, LeanContext makes the following contributions:

- It presents a low-cost domain-specific QA system, which reduces the LLM API usage cost by reducing the domain context (RAG context) through the selection of important sentences related to the query and keeping them intact, while reducing rest of the sentences in between important sentences through a free open-source summarizer. It proposes a reinforcement learning technique to select the percentage of the important sentences.
- It reduces the LLM API usage cost by $37.29\% \sim 67.81\%$ of a domain-specific QA system with little drop in performance by only $1.41\% \sim 2.65\%$. (Table 1, Table 2).
- Considering the RAG context based on a query as a baseline, LeanContext outperforms open-source summarizes by 6.19% to 15.5% while saving 39.25% of the overall cost (Table 3).
- It boosts the QA performance by $13.22\% \sim 24.59\%$ (Table 4) by combining query-aware top-k sentences with the reduced context generated through free open-source text summarizers.

2. Related work

For domain-specific tasks, LLMs can be utilized to adapt the domains without modifying their inner parameters via discrete prompting where distinct instructions with contexts can be delivered as an input to generate responses for downstream tasks (Ling et al., 2023; Brown et al., 2020). For domain-specific QA tasks, the domain can be reduced by context summarization to reduce the LLM cost. A lot of research has been conducted for summarizing text (Miller, 2019; Yang et al., 2023). Existing research works can be categorized into two main parts: (a) extractive and (b) abstractive. The extractive summarizers (Miller, 2019) first identify important sentences from the text, and next summarizes them. While abstractive summarizers (Laskar et al., 2023; Bhuyan et al., 2023) reduce the context by generating new sentences. The main goal of both approaches is to generate a meaningful summary for human users. In contrast, the goal of LeanContext is to reduce context which will be consumed by a question-answering model like ChatGPT. For the prompt-based summarization task, recently, iterative text summarization (Zhang et al., 2023b) has been proposed to refine the summary task in a feedback-based iterative manner. In aspect or query-based summarization (Yang et al., 2023) summaries are generated based on a domain set of specific queries.

Query-unaware text compression via prompting is also observed in recent literature. Semantic compression (Gilbert et al., 2023) involves generating systematic prompts to reduce context using ChatGPT model (GPT-3.5-turbo, GPT-4) and acquire reasonable compression compared to the zlib compression method. Due to limited context window size, recent literature focus on prompt context filtering. In selective context (Li,

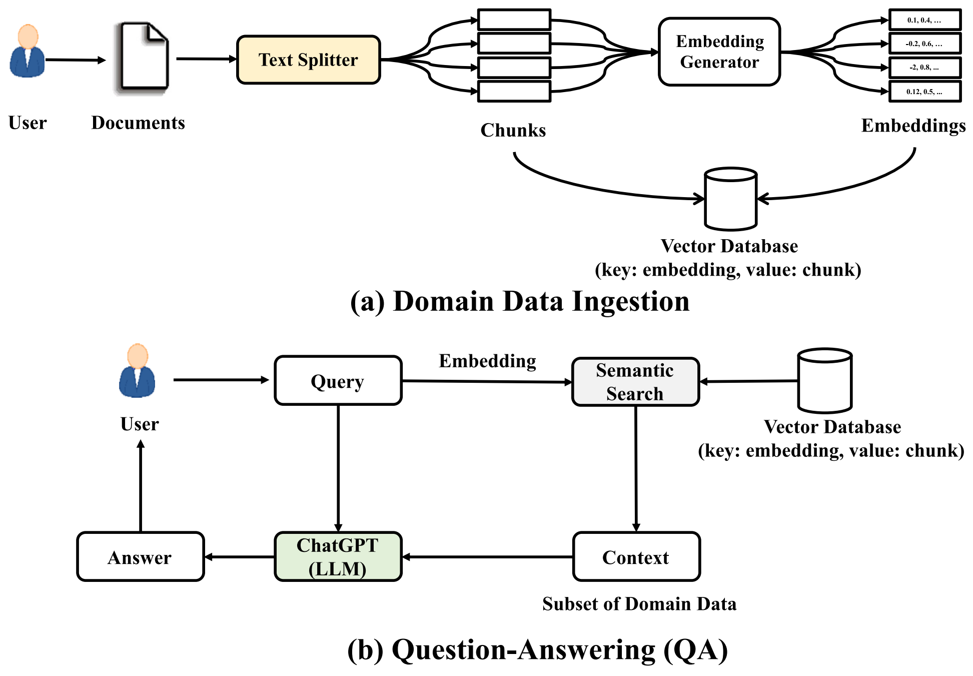


Fig. 2. Workflow of a RAG based domain-specific QA system.

2023), token, phrase, or sentence-level query-unaware content filtering is proposed using the entropy of GPT-2 model logits for each entity. Usage of ChatGPT model in the medical domain especially in radiology is explored via prompt-engineering (Ma et al., 2023) to summarize difficult radiology reports. Extract-then-generate pipeline-based summarization improves abstractive summary faithfulness (Zhang et al., 2023a) through the chain of thought (CoT) (Wei et al., 2022) reasoning. To reduce the cost of the use of LLM, FrugalGPT (Chen et al., 2023) proposed several ideas regarding prompt adaptation by query concatenation, LLM approximation by fine-tuning or caching, and LLM cascade by the selective selection of LLMs from lower to higher cost based on a query. Still, it lacks context compression ideas to reduce the prompt tokens.

It is to be noted that recent studies either focus on summarization as a downstream task or utilize the summary of the context for the question-answering task. For most of the existing content filtering approaches, the main focus is to query-agnostic filter content with the deletion of less informative content or for solely summarization tasks. Using LLM for query-aware context reduction adds extra overhead for using pay-per-use LLM to answer correctly. In addition, in recent articles, the chunk-based preprocessing of the article is ignored by assuming each content in the dataset as a chunk. In LeanContext, the main focus is to reduce the LLM cost by considering query-aware context reduction. Due to the possibility of rapid change of domain-specific user data, fine-tuning LLM or parameter-efficient LLM is not a feasible solution. Utilizing open-source LLM (Touvron et al., 2023; Chung et al., 2022) model either does not perform well on domain data or adds additional deployment cost. Consider a small business to run, we consider using pay-per-use LLMs such as OpenAI LLMs to make the system running at a reasonable cost by reducing the context.

3. Retrieval augmented generation

To implement a domain-aware question-answering (QA) system, LeanContext implements a traditional retrieval augmented generation (RAG) framework (Lewis et al., 2020). The RAG system is constructed through two distinct phases: (a) domain data ingestion and (b) query-response. Fig. 2 illustrates the workflow of a domain-aware QA system based on RAG.

3.1. Domain data ingestion

In this step, the documents, will be split into several fixed chunks (a chunk is a set of consecutive sentences in a document) by a text splitter. An embedding function computes the embeddings of each chunk using an embedding generator and generates an n-dimensional vector for each chunk. The chunks along with the embedding vector of each chunk are stored in a vector database (Anon, 2023; Pinecone, 2023). The vector database is useful in retrieving relevant chunks from the whole document based on the query which helps efficient processing of long documents.

3.2. Query-response

Given a user query, the embedding generator generates an n-dimensional vector representation of the query. A cosine similarity-based *semantic search* method determines N chunks by their embeddings similar to query embedding. These chunks form the context of the RAG system. Finally, the context with the query forms a prompt that is fed into LLM to get the response.

4. Challenges

As domain-specific data and user queries are dynamic, retrieving minimal context based on a query is challenging. One possible way is to make the chunk size and number of chunks dynamic so that the context contains minimal sentences to answer the query. But this solution is infeasible as the vector database needs to be reconfigured again per query with the change of domain. Instead, it will be more practical if after getting the possible chunks as context, the context is further reduced based on a query to get the near-optimal cost of LLM. Following this notion, we propose LeanContext, an adaptive context reduction system to reduce the prompt cost of ChatGPT-like LLMs.

5. LeanContext

5.1. Objective

The objective of LeanContext is to further reduce the context $\mathbf{C}$ to $\mathbf{C}'$ where the token count of $\mathbf{C}'$ is smaller than the token count of

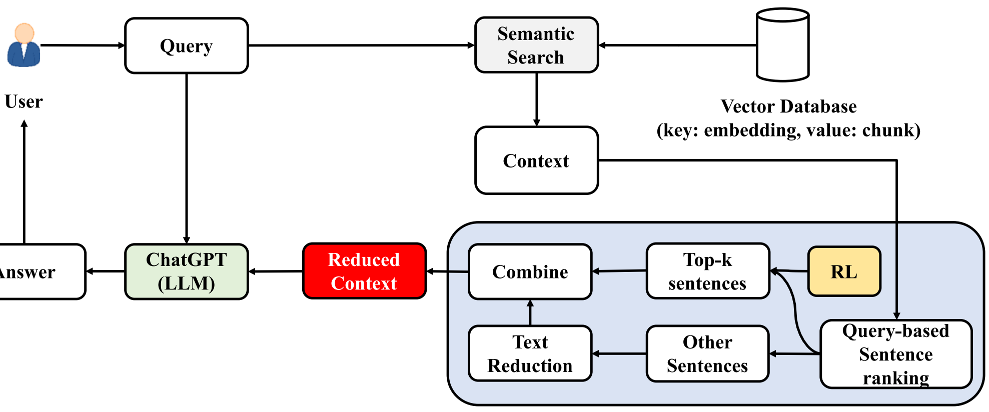


Fig. 3. System overview of LeanContext.

C. As LLM prompt cost is proportional to the token count of context, LeanContext helps to reduce the prompt cost of LLMs.

In other words, if the total number of tokens in **C** is T and the number of tokens in $\mathbf{C}'$ is t then LeanContext reduces the token ratio τ is defined as, $\tau = \frac{t}{T}$ without compromising the accuracy (acc) of the QA system. So, if the optimal accuracy of the system is acc^*, we formulate the optimization problem of LeanContext as follows.

$$\min. \quad (1-\alpha)\times\tau + \alpha\times|acc - acc^*| \tag{1}$$

Here, α controls the contribution of context ratio and accuracy. Finally, the reduced context $\mathbf{C}'$ and the query (q_i) are given to the pay-per-use LLM API to answer the query. Then, the answer is shown to the respective user via an interactive interface.

5.2. System design

For context-based QA, generally, the answers reside within a couple of sentences. If the smallest amount of context for a certain question can be identified, the same response can be provided by LLMs at a lower cost i.e. less prompt tokens. So, identifying the top-k sentences can reduce the context without compromising accuracy. Motivated by this simple idea, we propose LeanContext which is shown in Fig. 3.

After forming the context with semantic search, LeanContext first *ranks* the sentences of the context based on a query. Assuming the context consists of a sequence of sentences ($s_1, s_2, s_3, \ldots, s_n$), it extracts top-$k$ sentences similar to the query from context using the cosine-similarity function. To accomplish this, it computes the embedding of the query ($\mathbf{v}_q$), and the embedding is compared with each of the sentence embedding ($\mathbf{v}_{s_i}$) and *top-k sentences* are identified.

$$\text{Top-}k\text{ sentences} = sort(\mathbf{V}, \text{cosine\_similarity}(\mathbf{v}_q, \mathbf{v}_{s_i}))$$

where, $1 \le i \le n$.

The usage of only the top-k sentences as a simplified context is a cost-effective strategy to minimize the expenses associated with employing the Large Language Model (LLM) API. However, LeanContext takes this a step further by focusing on enhancing the accuracy of the information extraction process.

5.3. Construction of reduced context

Fig. 4 shows how LeanContext constructs the Reduced Context. It retains the most relevant top-k sentences, which are critical for preserving the core context, while concurrently applying an open-source text reduction method to condense the sentences that lie between these top-k sentences. This step helps in eliminating extraneous information and noise from the text. Moreover, any sentences that appear beyond the last top-k sentence are omitted from consideration. This selective approach ensures that only the most contextually relevant information is retained, contributing to a more efficient and focused analysis.

Importantly, LeanContext also maintains the original order of both the top-k sentences and the other sentences in the text. This preservation of sentence order ensures that the structural integrity of the input text remains intact. This holistic approach ultimately leads to more accurate and informative results when processing text data, while still achieving the goal of minimizing LLM API usage costs.

The idea of preserving the top-k sentences in their original form is a straightforward yet compelling approach. Incorporating this simple approach into existing open-source pre-trained summarization models has the potential to enhance their performance. However, a challenging question arises when it comes to determining the value of k when selecting top-k sentences. The optimal value of k varies based on the query and context. We pose this question: *What is the appropriate value for k?*. To provide a more detailed answer, we present a solution based on reinforcement learning (RL) as follows.

5.4. Reinforcement learning for adaptive-k

Identifying the number k is crucial for context reduction as well as performance. Considering a fixed-k might impact accuracy if $k \ll n$. On the contrary, if $k \simeq n$, the token ratio will be higher, leading to higher costs. Thus, to achieve minimal cost with maximum accuracy, the k should be adaptive. To get a query-based adaptive context, we propose a lightweight Q-learning-based reinforcement learning algorithm that perceives an optimal policy for an agent operating in an environment. After training, we will have an optimal Q table that takes the best action for a given state.

$$\pi^*(state) = \underset{a}{\text{argmax}}\ Q^*(state, action)$$

5.5. Training algorithm

The off-policy Q table training algorithm is shown in Algorithm 1. In line 1–6, each state is computed by the subtraction of query embedding from context embedding. A k-means model is trained to get the centroids and utilize the centroids as different states. In line 7–15 for selecting each threshold as action, the corresponding reward is computed the Q table is updated. Finally, the updated Q table is deployed for reducing the context. The training of RL requires LLM to compute the reward. To reduce the training cost, we perform training on fewer samples. Hence to observe the effect of each action in each state, we do a full exploration to update the Q table.

5.6. RL components

Due to the variation of domains and user queries, it is difficult to estimate the fixed context size. In this situation where the environment is complex and dynamic, RL fits the best. After retrieving the context based on a query, LeanContext computes the state by subtracting the

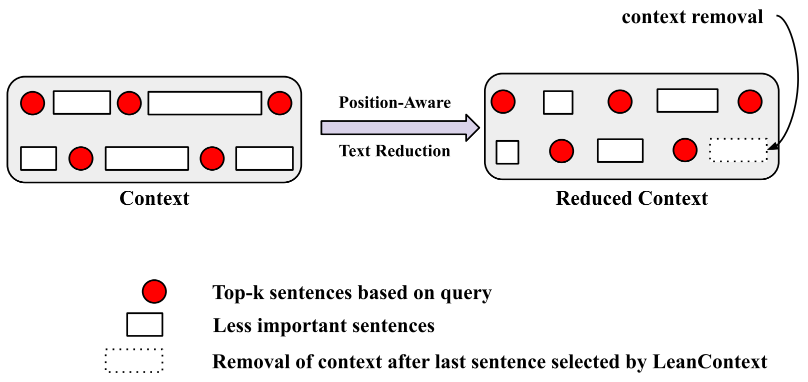

context removal
Position-Aware
Text Reduction
Context
Reduced Context
Top-k sentences based on query
Less important sentences
Removal of context after last sentence selected by LeanContext

Fig. 4. Construction of reduced context in LeanContext.

Algorithm 1 LeanContext Training Algorithm

```
Require: $\mathcal{D}$: Input documents
Require: $q$: A set of queries
Require: $\Theta$: a set of predefined thresholds, acts as a set of actions in RL
Ensure: $Q^*$: Trained $Q$-table
 1: $\mathcal{X} \leftarrow \emptyset$                                  ▷ set of states
 2: for each $q, context \in q, \mathcal{D}$ do
 3:     $\mathbf{v}_c, \mathbf{v}_q \leftarrow$ Embedding($context$), Embedding($q$)     ▷ get embedding vectors
 4:     $\mathcal{X} \leftarrow \mathcal{X} \cup (\mathbf{v}_c - \mathbf{v}_q)$
 5: end for
 6: $\mathcal{S} \leftarrow K\text{-}Means(\mathcal{X})$                 ▷ centroids as state vectors
 7: for each $q, context \in q, \mathcal{D}$ do
 8:     $state \leftarrow$ get_state($\mathcal{S}, context, q$)
 9:     $action \leftarrow$ get_action($\Theta$)
10:     $\mathbf{C} \leftarrow$ retrieve($q, \mathbf{v}_q, \theta$)               ▷ context formation
11:     $\mathbf{C}' \leftarrow$ perform_action($\mathbf{C}, action$)          ▷ text reduction
12:     $answer \leftarrow llm(q, \mathbf{C}')$
13:     $r \leftarrow$ compute_score($answer, original\_answer$)
14:     $reward \leftarrow \alpha(2r - r^*) - (1-\alpha) \times \tau(\mathbf{C}', \mathbf{C})$
15:     $Q(state,action) \leftarrow Q(state,action) + \frac{1}{n}(reward - Q(state,action))$   ▷ Update $Q$ table
16: end for
17: return $Q^*$                                  ▷ return trained $Q$ table
```

query embedding from the context embedding. With the state, the RL agent finds a suitable action from the trained Q^* table. Based on the action, the threshold for context reduction is computed and top-k sentences are selected. After that the response is retrieved from the reduced context and query and sent to the LLM API to get the response. We carefully define our state, action, and reward function as follows.

5.6.1. State

We combine query and context to define the *state*. At offline profiling, we compute the embedding of the query $\mathbf{v}_q$ and the embedding of the context as $\mathbf{v}_c$. Then we subtract $\mathbf{v}_q$ from $\mathbf{v}_c$ that indicates the context-query pair. The rationale behind this is explained in detail in Section 9. After building the vector with several training samples, we run the *K-means* model to compute the centroids. These centroids are utilized as state vectors $\mathcal{S}$. The number of clusters should be determined in a way so that the similar representation state vector generated from the context-query pair selects the same k sentences from the context. Based on the action value range stated below, we select the number of clusters to 8.

$$\mathcal{S} \leftarrow K - means\left(\bigcup_{i,j}(\mathbf{v}_{c_i} - \mathbf{v}_{q_j})\right)$$

5.6.2. Action

We aim to develop an adaptive top-k sentences extraction method. To achieve this, we establish a range of thresholds, varying from 0.05 to 0.4 (where the maximum top-k limit is set at 40% of the total context), which are used to determine the number of sentences to be retained. The k in top-k is derived from the current action value and the total number of sentences in the context. If the current action value is a and the total number of sentences in the context is n, then $k = a \times n$. Each potential choice within this range is considered an action in the proposed reinforcement learning (RL) system. The outcome of each action is evaluated through a reward function, which subsequently updates the Q table for the corresponding (state, action) pair.

LeanContext retains the top-k sentences while shortening the less important sentences between them and discards all sentences that are less important beyond the last top-k sentence. Therefore, the extent of context reduction depends on the length of these less significant sentences. To ensure that these sentences maintain a substantial length, the RL agent limits the maximum top-k ratio to 0.4.

5.6.3. Reward

The reward for the RL model is higher if the context ratio is less and the accuracy is almost equal to the accuracy using the original context. We compute the ROUGE score r to evaluate the answer using reduced context with the actual answer using the original context. If the ROUGE score with original context is r^* for a query, then the current (state, action) will be rewarded if $r - r^* \geq 0$, otherwise will be penalized. For the token ratio τ, the lower the better as the reduction of context as much as possible without compromising accuracy is rewarded. Thus, the reward function $\mathbf{R}$ is defined as follows.

$$\mathbf{R} = -(1-\alpha)\tau + \alpha(2r - r^*)$$

Here, α controls the significance of context reduction and accuracy on the reward function. α is set to 0.9 in all experiments.

5.7. LeanContext inference

The LeanContext inference algorithm is shown in Algorithm 2. For each query, the corresponding context is retrieved, and the state is computed by the trained RL-agent [line 2]. Based on the state, the threshold θ is computed as an action to select the top-k sentences and less important sentences. Among the less important sentences, sentences before last index from top-k sentence set are reduced and the rest less important sentences after top-k are eliminated. Thus the reduced context, $\mathbf{C}'$ is produced [line 3–4]. This reduced context is utilized to get answer with the LLM [line 5]. Finally, the answer is returned to the user.

Algorithm 2 LeanContext Inference

```
Require: $\mathcal{D}_t$: Test documents
Require: $q_t$: A set of test queries
Require: $Agent$: Trained RL-Agent
1: for each $q, context \in q_t, \mathcal{D}_t$ do
2:     $state \leftarrow Agent$.get_state($Agent.\mathcal{S}, context, q$)
3:     $action \leftarrow Agent$.get_action($state$)
4:     $\mathbf{C}' \leftarrow Agent$.perform_action($context, action$)
5:     $answer \leftarrow llm(q, \mathbf{C}')$
6:     return $answer$
7: end for
```

6. Experimental settings

6.1. Dataset

In real-time scenarios, user documents are new to LLMs such as gpt-3.5-turbo model, and thus should not be able to answer a query without giving a context. To ensure this, we use recent arxiv papers and BBC news articles so that the LLMs are not trained on these. Following this, we use the Arxiv Dataset and BBCNews Dataset where documents are published in March 2023 (Li, 2023). For comparison with the existing datasets, we also evaluate LeanContext on NarrativeQA (Kočiskỳ et al., 2018). For BBC News, and Arxiv datasets, we generate questions with answers for each document using QAGenerationChain from Chase (2022) based on gpt-3.5-turbo model.

6.2. Baseline models

In our question-answering-based system, we mainly focus on reducing the context length while keeping the same performance in QA. So, we use a pay-per-use LLM (`gpt-turbo-3.5`) model for all cases to answer a query based on the context. We evaluate our proposed context length reduction approach with the recent context reduction approaches as follows.

1. Context (Original): We keep the context length intact and ask LLM to answer the query.
2. CQSumDP (Laskar et al., 2023): We generate the following prompt similar to CQSumDP to generate the query-aware summary of a context.
   "A document along with its query is given below. Write down the most reasonable summary relevant to its document-query pair.
   Document: {CONTEXT}
   Query: {QUERY}"
3. Semantic Compression (Gilbert et al., 2023): A query unaware prompt is stated to compress a context as follows.
   "Please compress the following text into a latent representation that a different `gpt-3.5-turbo` model can decompress into the original text. The compression model should purely minimize the number of characters in the compressed representation while maintaining the semantics of the original text. The resulting compressed text does not need to be decompressed into the original text but should capture the semantics of the original text. The compressed text should be able to be decompressed into a text that is semantically similar to the original text but does not need to be identical.
   Text to Compress: {CONTEXT}"
4. SBert (Miller, 2019): A Bert-model-based extractive summarization approach. The context is reduced to several sentences. In our experiments, we set the number of sentences to 3.
5. Selective Context (SC Li, 2023): It utilizes entropy to filter out less informative content from context. In our experiments, we employ phrase-level content filtering with different reduction ratios. We use GPT-2 model to compute the self-information.
6. Flan-T5-Base (Chung et al., 2022): We use this 250M encoder–decoder model to summarize the context based on the same instruction as CQSumDP.

6.3. Implementation details

To implement the document ingestion, we use a cheap all-MiniLM-L6-v2 model (Reimers and Gurevych, 2019) as an embedding generator. The text chunks along with the embeddings are stored in ChromaDB (Anon, 2023) vector database. We use the chunk size 500, chunk overlap is 0. We vary the number of chunks $N = 2, 4, 8, 10$ to compare how well the RL algorithm performs. We use this template for the QA: *"Answer to the question based on the given context. Context: {CONTEXT}, Question: {QUERY}, if you do not find any answer in the context, simply return 'No answer'"*. We use LLMChain from Chase (2022) to ask queries to the OpenAI model by prompting. LeanContext uses selective context method (Li, 2023) to reduce less important sentences by 80%.

7. Results

7.1. Arxiv dataset

We consider random 25 articles. As we choose new articles for evaluation, we need to generate the questions and answers from these documents to evaluate LeanContext. Due to the cost of LLM API usage and the cost limit for API usage set by OpenAI (OpenAI-Policy, 2023), we follow recent literature for the experimental setup (Yang et al., 2023). We have used 100 queries per dataset to evaluate LeanContext. For the Arxiv dataset, we chose 4 questions from each paper, resulting in a total of 25 papers selected for testing. The total number of sentences in these documents varied from 63 to 962 with an average of 352 sentences per document. We use another 5 documents for the RL agent training distinct from the 25 documents. We train the agent on 83 context, questions, and answer triples. To evaluate the samples on almost each action value, we have run the agent on the query 8 times (number of action values), totaling 664 epochs. For evaluation, after retrieving the query, we use the RL agent to identify to top-k in the context and reduce the context using the RL agent. For the same query, and context setting, we evaluate our approach with the baseline approaches. The comparison of LeanContext with baseline models is shown in Table 1. LeanContext achieves similar performance compared to the original context with no text reduction and outperforms existing open-source models with 37% cost savings. CQSumDP (Laskar et al., 2023) achieves better accuracy with a greater cost (adds 15.36% more cost) by reducing the minimal context to get the right answer than the original context. Throughout the experiments, we observe the same effect and conclude that the zero-shot performance of ChatGPT like LLMs is better with concise and relevant context to query (CQSumDP) than the context with a lot of irrelevant information. Although Semantic Compression (Gilbert et al., 2023) uses LLM to minimize context, due to the context-agnostic nature of the prompt, it performs even worse than LeanContext and adds additional $\sim 72\%$ cost.

7.2. BBCNews dataset

For the same limitation of OpenAI API usage stated above, we consider random 100 news articles from the BBCNews dataset. The total number of sentences in these documents varied from 4 to 139, with an average of 30 sentences per document. We keep 80 articles for testing and the rest 20 articles to train the RL agent. From the 20 articles, we generate 27 query, context, and answer triples, and similar to the Arxiv dataset, we run the experiment for 216 epochs to give a reward or penalty to almost each action value. Finally, for query-based context retrieval settings, we evaluate our approach with the baseline approaches.

Table 1
Comparison on Random 100 samples from Arxiv Dataset. Number of chunks, $N = 4$.

Text reduction method	Avg. Total tokens	Avg. Prompt tokens	Avg. Summary tokens	Avg. Completion tokens	ROUGE-1	ROUGE-2	ROUGE-L	Cost savings (%)
Context (Original)	547	521	0	26	**0.3985**	**0.2868**	**0.3714**	0.00
CQSumDP	631	89	517	25	**0.4424**	**0.3061**	**0.4213**	**−15.36**
Semantic Compression	939	319	597	23	0.3331	0.2221	0.3132	−71.66
T5-base	79	71	0	8	0.1614	0.1101	0.1486	85.56
SBert	205	188	0	17	0.2563	0.1701	0.2469	62.52
SC (reduction = 0.50)	334	316	0	18	0.2945	0.2014	0.2755	38.94
LeanContext (Fixed k =0.1)	210	196	0	14	0.2305	0.1623	0.2173	61.62
LeanContext (Adaptive k [RL])	**343**	**321**	**0**	**22**	**0.3844**	**0.2684**	**0.3577**	**37.29**

Table 2
Comparison on Random 100 samples from BBCNews Dataset. Number of chunks, $N = 8$.

Text reduction method	Avg. Total tokens	Avg. Prompt tokens	Avg. Summary tokens	Avg. Completion tokens	ROUGE-1	ROUGE-2	ROUGE-L	Cost savings (%)
Context (Original)	761	724	0	37	**0.5498**	**0.4172**	**0.5337**	0.00
CQSumDP	842	75	738	29	**0.5801**	**0.4405**	**0.5637**	**−10.64**
Semantic Compression	1078	228	820	30	0.4729	0.3204	0.4517	−41.66
T5-base	74	51	0	23	0.3993	0.2631	0.3752	90.28
SBert	174	147	0	27	0.4261	0.2917	0.4082	77.14
SC (reduction = 0.50)	461	429	0	32	0.4740	0.3308	0.4521	39.42
LeanContext (Fixed k = 0.1)	**278**	**250**	**0**	**28**	**0.5017**	**0.3740**	**0.4872**	**63.47**
LeanContext (Adaptive-k [RL])	**245**	**218**	**0**	**27**	**0.5233**	**0.3943**	**0.5093**	**67.81**

Table 3
Comparison on 500 queries from NarrativeQA Dataset (Kočiský et al., 2018). Number of chunks, $N = 4$. Response with original context is considered as ground truth for this experiment.

Text reduction method	Avg. Total tokens	Avg. Prompt tokens	Avg. Completion tokens	ROUGE-1	ROUGE-2	ROUGE-L	Cost savings (%)
Context (Original)	456	391	65	GT	GT	GT	0.00
T5-base	114	57	57	0.3694	0.2169	0.3459	75.00
SBert	205	142	63	0.4625	0.2937	0.4338	55.04
LeanContext (top-$k = a \times n$)	**230**	**168**	**62**	**0.4952**	**0.3309**	**0.4678**	**49.56**
LeanContext	**277**	**220**	**57**	**0.5244**	**0.3621**	**0.4967**	**39.25**

Due to the cost of the OpenAI model and the current usage limit, we follow recent literature (Yang et al., 2023), we generate queries using the QA generation method, and consider random 100 query samples for evaluation. The evaluation result is shown in Table 2.

We observe that generating a query-aware summary from document-query pair using OpenAI LLM (Laskar et al., 2023) performs better i.e. ROUGE-1 0.5801 than query-aware open-source baseline methods such as T5-base i.e. 0.3993 but with a high cost. However, it also contributes more cost than the original context (10.64% more). We also observe that the query-unaware LLM using a different hard prompt (Gilbert et al., 2023) even performs worse i.e. ROUGE-1 score 0.4729 with a 41.66% cost overhead. Additionally, adding fixed top-k (where $k = 0.1 \times n$, n is the total number of sentences in the context) sentences with the summarization of the original context using T5-base model uplifts the ROUGE-1 score of T5-base by $\sim$ 12.35%. With only adaptive top-k sentences computed by RL method ($k = a \times n$, a is the action value) and adaptive top-k ($k = a \times n$) sentences along with sentence order (our LeanContext), it performs even better by reducing the cost from 65% $\sim$ 74%. We observe the same scenario for other models too. Further investigation with different numbers of chunks for the same test queries is discussed in Section 9.

7.3. NarrativeQA dataset

For NarrativeQA dataset (Kočiský et al., 2018), we select the stories from the train data containing the first 200 queries, we concatenate the stories to form a long train document. The document is converted into chunks. Using the embedding generator, the embedding for each chunk is generated and the chunks with the chunk embeddings are stored in a vector database for query-based retrieval of chunks. Thus, we train our RL agent with the 200 queries. For test queries, we follow the same approach in building the test long document except for the number of queries. We use 500 queries to evaluate LeanContext with baseline models. We set the chunk size to 500 and the chunk overlap to 0 for both training and testing. For a given query, we extract the top 4 chunks from the database and form a context. Then, we build the prompt using the context and query and give it to LLM for the response. We use this response as ground truth to compare with the baselines.

The results are illustrated in Table 3. We observe that LeanContext achieves 15.5% more ROUGE-1 score than T5. For SBert, LeanContext outperforms by 6.19%. Even with only top-k sentences chosen by LeanContext, the ROUGE-1 score is higher than SBert and T5-base. Although the SBert and T5 models achieve more cost savings, they also suffer from low accuracy. Overall, our LeanContext saves 39.25% cost.

7.4. Top-k sentences is all you need

We observe an interesting phenomenon when adding our LeanContext with existing open-source models. In Table 4, we compare the results of open-source summarizer models by adding top-k sentences identified by LeanContext with the summarized text by each summarizer model (Sbert, T5) as a context.

As these open-source models are not trained on new domain data, we observe that combining only 10% top-k ($k = 0.1 \times n$) sentences with the open-source models [SBert + LeanContext (top-$k = 0.1 \times n$), T5 + LeanContext (top-$k = 0.1 \times n$)] boosts the QA system's performance by 5.41% $\sim$ 17.11% for the Arxiv dataset and 8.11% $\sim$ 12.35% for the BBCNews dataset as shown in Table 4. Here, n is the total sentences in

Table 4
Effect of cascading LeanContext with open source summarizers.

Dataset	Text Reduction Method	Avg. Total tokens	Avg. Prompt tokens	Avg. Summary tokens	Avg. Completion tokens	ROUGE-1	ROUGE-2	ROUGE-L	Cost savings (%)
Arxiv	**Context (Original)**	**547**	**521**	**0**	**26**	**0.3985**	**0.2868**	**0.3714**	**0.00**
	T5	79	71	0	8	0.1614	0.1101	0.1486	85.56
	T5 + LeanContext (top-k=0.1 × n)	131	113	0	18	0.3325	0.2390	0.3146	76.05
	T5 + LeanContext (top-k=a × n)	284	263	0	21	0.3942	0.2847	0.3742	48.08
	T5 + LeanContext	**357**	**335**	**0**	**22**	**0.4073**	**0.2863**	**0.3809**	**34.73**
	SBert	205	188	0	17	0.2563	0.1701	0.2469	62.52
	SBert + LeanContext (top-k=0.1 × n)	250	230	0	20	0.3104	0.2181	0.2949	54.30
	SBert + LeanContext (top-k=a × n)	405	380	0	25	0.3676	0.2594	0.3464	25.96
	SBert + LeanContext	**478**	**452**	**0**	**26**	**0.3885**	**0.2720**	**0.3596**	**12.61**
BBCNews	**Context (Original)**	**761**	**724**	**0**	**37**	**0.5498**	**0.4172**	**0.5337**	**0**
	T5	74	51	0	23	0.3993	0.2631	0.3752	90.28
	T5 + LeanContext (top-k=0.1 × n)	142	116	0	26	0.5228	0.3914	0.5065	81.34
	T5 + LeanContext (top-k=a × n)	192	165	0	27	0.5368	0.4072	0.5187	74.77
	T5 + LeanContext	**259**	**232**	**0**	**27**	**0.5532**	**0.4298**	**0.5374**	**65.97**
	SBert	174	147	0	27	0.4261	0.2917	0.4082	77.14
	SBert + LeanContext (top-k=0.1 × n)	241	212	0	29	0.5072	0.3731	0.4914	68.33
	SBert + LeanContext (top-k=a × n)	289	260	0	29	0.5200	0.3827	0.5039	62.02
	SBert + LeanContext	**355**	**327**	**0**	**28**	**0.5355**	**0.4007**	**0.5235**	**53.36**

Table 5
Evaluation of RL model on a different number of chunks on a random 100 samples from BBCNews Dataset. Our RL agent trained with (N=8) shows promising results while applying on different chunk numbers.

Text reduction method	Avg. Total tokens	Avg. Prompt tokens	Avg. Summary tokens	Avg. Completion tokens	ROUGE-1	ROUGE-2	ROUGE-L	Cost savings (%)
Number of chunks, N=2								
Context (Original)	243	211	0	32	0.5370	0.3987	0.5190	0.00
CQSumDP	322	70	225	27	0.5303	0.3897	0.5105	–32.51
Semantic Compression	448	113	307	28	0.4786	0.3169	0.4519	–84.36
T5-base	76	50	0	26	0.4010	0.2640	0.3781	68.72
SBert	157	128	0	29	0.4825	0.3395	0.4604	35.39
SC (reduction = 0.50)	164	137	0	27	0.4614	0.3113	0.4403	32.51
LeanContext	117	92	0	25	**0.4556**	**0.3265**	**0.4373**	**51.85**
Number of chunks, N=4								
Context (Original)	417	384	0	33	0.5489	0.4183	0.5327	0.00
CQSumDP	496	71	398	27	0.5569	0.4156	0.5389	–18.94
Semantic Compression	665	159	479	27	0.4916	0.3328	0.4684	–59.47
T5-base	74	50	0	24	0.3921	0.2555	0.3678	82.25
SBert	163	136	0	27	0.4626	0.3287	0.4452	**60.91**
SC (reduction = 0.50)	263	235	0	28	0.4726	0.3250	0.4507	36.93
LeanContext	153	128	0	25	**0.4856**	**0.3605**	**0.4700**	**63.31**
Number of chunks, N=10								
Context (Original)	930	892	0	38	0.5423	0.4072	0.5246	0.00
CQSumDP	1008	74	906	28	0.5849	0.4440	0.5644	–8.39
Semantic Compression	1274	258	987	29	0.4862	0.3343	0.4642	–36.99
T5-base	74	51	0	23	0.3903	0.2584	0.3688	92.04
SBert	181	153	0	28	0.4164	0.2807	0.3967	**80.54**
SC (reduction = 0.50)	559	527	0	32	0.4825	0.3312	0.4591	39.89
LeanContext	279	252	0	27	**0.5318**	**0.4069**	**0.5205**	**70.00**

the context. SBert + LeanContext (top-k = 0.1 × n) means we compute the summarized text of the original context by SBert and add the important 10% sentences of the same original context identified by LeanContext with it to observe the improvement. In addition, using only top-k sentences with RL method (k = a×n) from LeanContext with a generic summarizer adds an extra advantage to the overall system [SBert + LeanContext (top-k = a×n), T5 + LeanContext (top-k = a×n)]. Here, a is the action value (0.05–0.4) that indicates the top-k ratio. Our LeanContext (combining with top-k sentences using RL and reduced versions of other sentences by preserving sentence order) makes the performance even better than the fixed (10%) top-k by 13.22% ∼ 24.59% for the Arxiv dataset and 10.94% ∼ 15.39% for the BBCNews dataset (Table 4). LeanContext with T5-base model outperforms all the existing approaches including no reduction method.

8. Human evaluation

For human evaluation on queries, we evaluate the responses using the original context and the baseline models with LeanContext for the NarrativeQA dataset (Kočiskỳ et al., 2018). We observe that T5 and SBert either offer incorrect responses (Query #1 and #4) or provide partial responses (Query #2 and #3).

1. **Query:** In what valley did the solemn revels of Cynthia take place?
   Response:
   - **Original Context:** The solemn revels of Cynthia took place in the Valley of Gargaphie.
   - **LeanContext:** The solemn revels of Cynthia took place in the Valley of Gargaphie.

- **T5:** The solemn revels of Cynthia took place in the valley of Nysa.
- **SBert** The solemn revels of Cynthia took place in the Valley of Gargaphie.

2. **Query:** What did the symbolic vices disguise themselves to be?
   Response:

   - **Original Context:** The symbolic vices disguised themselves as delight and virtue. In the context provided, vice is portrayed as a force that can captivate the vicious mind, transforming what is initially perceived as delight into punishment. This suggests that vices often present themselves in appealing or virtuous guises, misleading individuals to engage with them under the false pretense of pursuing something good or pleasurable. Additionally, the reference to Romola's perception of Savonarola and the Church indicates that vices can also masquerade as moral or spiritual authority, leading individuals astray by exploiting their trust and faith in what they believe to be virtuous or righteous paths.
   - **LeanContext:** The symbolic vices disguised themselves as delight and greatness, but in reality, they led to punishment and degradation, sinking beauty and greatness into vice and rags. They operated under the guise of virtues or desirable qualities, misleading individuals into believing in their positive nature while actually leading them towards moral and spiritual downfall.
   - **T5:** The symbolic vices disguised themselves as virtues.
   - **SBert** The symbolic vices disguised themselves as delight or pleasure to the vicious mind, turning what is perceived as delight into punishment.

3. **Query:** Who challenges the courtiers to court compliment competition?
   Response:

   - **Original Context:** Amorphus challenges the courtiers to a court compliment competition.
   - **LeanContext:** ANA challenges the courtiers to a court compliment competition at the Bare Accost.
   - **T5:** In the context of "ANA", it is Ana who challenges the courtiers to a court compliment competition.
   - **SBert** Based on the provided context, there is no explicit mention of anyone challenging the courtiers to a court compliment competition. The text discusses the virtue of true nobility, the critique of certain courtiers' behavior, and a brief mention of a legal situation involving an appeal and the friends of the accused. Therefore, based on the given context, it is not possible to determine who challenges the courtiers to a court compliment competition.

4. **Query:** Who does Diana represent?
   Response:

   - **Original Context:** Diana represents the Roman goddess of the hunt, the moon, and nature, associated with wild animals and woodland, and having the power to talk to and control animals. She is also known for her chastity and is often depicted as a huntress carrying a bow and arrows. In the context provided, Diana's representation might also symbolize purity, virtue, and a form of divine authority or judgment, especially in the way her court and the characters around her are described, focusing on themes of chastity, admiration, and divine will.
   - **LeanContext:** Diana represents the Roman goddess of the hunt, the moon, and nature, associated with wild animals and woodland, and having the power to talk to and control animals. She is equivalent to the Greek goddess Artemis in mythology. In the context provided, Diana's ability to "distinguish times" and "sort her censures" while keeping divine judgments to herself, suggests her wisdom and authority, characteristics befitting her mythological role.
   - **T5:** Diana represents the star of Cynthia's court.
   - **SBert** Diana represents the Roman goddess of the hunt, the moon, and nature, associated with wild animals and woodland, and having the power to talk to and control animals. She is known for her virginity and is a protector of chastity and a nurturer of the young. Diana is often depicted as a huntress carrying a bow and arrows. In the context provided, Diana is invoked as a symbol of purity and natural beauty, embodying the ideals of virginity and chastity.

5. **Query:** What injury does Micky receive as he attempted to help his step-brother in a fight with police?
   Response:

   - **Original Context:** Micky receives an injury where his head is split open and bleeding as he defends himself from the police beating him with their flashlights.
   - **LeanContext:** Micky receives a head injury, specifically his head is split open and bleeding, as he attempts to help his step-brother in a fight with the police.
   - **T5:** Micky receives a head injury, specifically his head is split open and bleeding, as he attempted to help his step-brother in a fight with police.
   - **SBert** Based on the provided context, there is no specific mention of Micky receiving an injury while attempting to help his step-brother in a fight with police. The context focuses on a scene involving Micky and Dicky, presumably in a boxing ring, with no direct reference to a fight with police or an injury sustained by Micky in such a scenario.

9. Ablation study

9.1. Variable number of chunks

We investigate whether the RL model has a performance dependency on the number of chunks. So, we train the RL model using $N = 8$ and run the same model on $N = 2, 4$, and 10. The results are shown in Table 5. We observe that although the N changes, the RL model still outperforms other open source baseline methods. Another interesting observation lies in making N as larger as possible for the successful retrieval of context using LeanContext whereas previously without LeanContext N should be kept smaller to reduce the cost of LLM usage.

9.2. Distribution of k

In Fig. 5, we show how the action (top-k ratio) is taken by the RL agent given the context and query for each query sample out of 100 samples. Considering that the adaptive top-k ratio chosen by our RL agent varies across query samples, this achieves an adaptive reduction of context.

For the Arxiv Dataset, the total number of sentences in the RAG context varied from 9 to 25 with an average of 15 sentences per context. Similarly, for BBCNews dataset, the total number of sentences in the context varied from 18 to 34, with an average of 26 sentences per context. For the Arxiv and BBCNews dataset, the distribution of top-k sentences over 100 queries for each dataset using LeanContext is shown in Fig. 6 and Fig. 7, respectively.

We also empirically evaluate state function with different associations between context ($\boldsymbol{v}_c$) and query embedding ($\boldsymbol{v}_q$) i.e. cosine

Table 6
Comparison of RL state functions on Arxiv dataset.

Compression method	Avg. Total tokens	Avg. Prompt tokens	Avg. Completion tokens	ROUGE-1	ROUGE-2	ROUGE-L
None	538	514	24	0.3945	0.2904	0.3764
LeanContext $\left(\text{state} = \frac{\boldsymbol{v}_c \cdot \boldsymbol{v}_q}{\|\boldsymbol{v}_c\|\|\boldsymbol{v}_q\|}\right)$	296	275	21	0.3443	0.2369	0.3259
LeanContext (state = $\boldsymbol{v}_c - \boldsymbol{v}_q$)	331	308	23	0.3553	0.2535	0.3388
LeanContext (state = $\boldsymbol{v}_c \oplus \boldsymbol{v}_q$)	284	265	19	0.3400	0.2370	0.3223

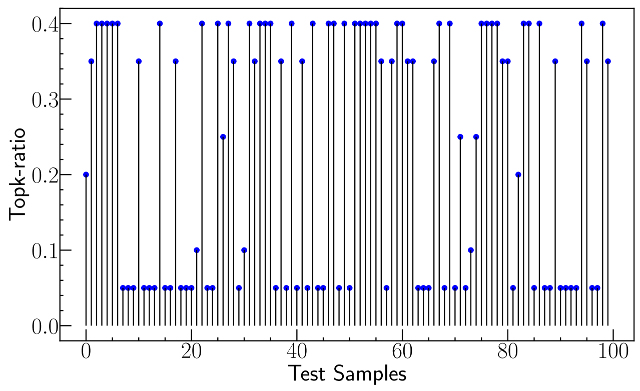


Fig. 5. Adaptive-k ratio selected by the BBCNews RL agent based on queries.

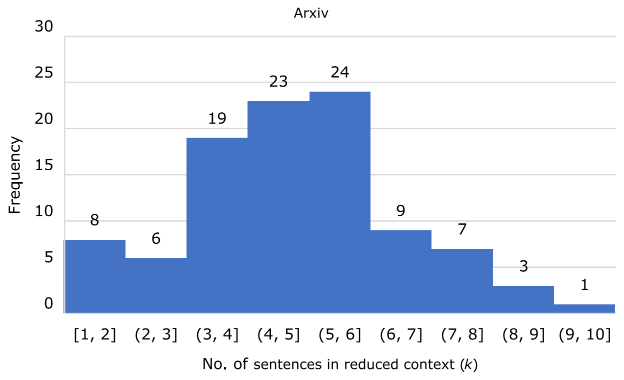


Fig. 6. Distribution of k for Arxiv data.

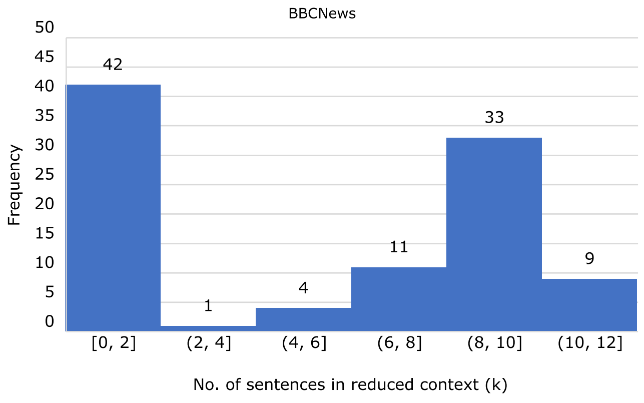


Fig. 7. Distribution of k for BBCNews data.

similarity $\left(\text{state} = \frac{\boldsymbol{v}_c \cdot \boldsymbol{v}_q}{\|\boldsymbol{v}_c\|\|\boldsymbol{v}_q\|}\right)$, concatenation ($\boldsymbol{v}_c \oplus \boldsymbol{v}_q$), and subtraction ($\boldsymbol{v}_c - \boldsymbol{v}_q$). Among them, subtraction performs best (Table 6). In this experiment, we only select the top-k sentences using RL to evaluate the impact of state definition on performance.

10. Conclusion and future work

In this paper, we introduce LeanContext, a cost-efficient, query-aware context reduction system designed to reduce the costs associated with LLM API usage. Despite the reduction in prompt tokens, LeanContext achieves similar or superior performance compared to maintaining the original context without reduction. The primary advantage of employing the top-k approach lies in its compatibility with any existing summarization method within a domain-aware question-answering system, thereby enhancing overall performance, as demonstrated by our experimental results. For the scope of this study, our emphasis is solely on text-based context within a domain. Future work will extend our exploration to other domains.

CRediT authorship contribution statement

Md Adnan Arefeen: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Biplob Debnath:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Investigation, Formal analysis, Conceptualization. **Srimat Chakradhar:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Investigation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Biplob Debnath, Srimat T. Chakradhar, Md Adnan Arefeen has patent pending to NEC Labs America Inc. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Anon, 2023. ChromaDB. https://www.trychroma.com/. (Accessed: 20 June 2023).
Bhuyan, S.S., Mahanta, S.K., Pakray, P., Favre, B., 2023. Textual entailment as an evaluation metric for abstractive text summarization. Nat. Lang. Process. J. 4, 100028.
Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al., 2020. Language models are few-shot learners. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901.
Chase, H., 2022. LangChain. https://github.com/hwchase17/langchain. 2022-10-17.
Chen, L., Zaharia, M., Zou, J., 2023. FrugalGPT: How to use large language models while reducing cost and improving performance. arXiv preprint arXiv:2305.05176.
Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al., 2022. Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416.
Espejel, J.L., Ettifouri, E.H., Alassan, M.S.Y., Chouham, E.M., Dahhane, W., 2023. GPT-3.5, GPT-4, or BARD? Evaluating LLMs reasoning ability in zero-shot setting and performance boosting through prompts. Nat. Lang. Process. J. 5, 100032.
Gilbert, H., Sandborn, M., Schmidt, D.C., Spencer-Smith, J., White, J., 2023. Semantic compression with large language models. arXiv preprint arXiv:2304.12512.
GPT-3 Cost, 2023. GPT-3 cost estimation for real applications. URL https://neoteric.eu/blog/how-much-does-it-cost-to-use-gpt-models-gpt-3-pricing-explained/. (Accessed: 11 July 2023).
Jiao, W., Wang, W., Huang, J.-t., Wang, X., Tu, Z., 2023. Is ChatGPT a good translator? A preliminary study. arXiv preprint arXiv:2301.08745.

Kočiskỳ, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K.M., Melis, G., Grefenstette, E., 2018. The narrativeqa reading comprehension challenge. Trans. Assoc. Comput. Linguist. 6, 317–328.

Laskar, M.T.R., Rahman, M., Jahan, I., Hoque, E., Huang, J., 2023. CQSumDP: A ChatGPT-annotated resource for query-focused abstractive summarization based on debatepedia. arXiv preprint arXiv:2305.06147.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., Kiela, D., 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20.

Li, Y., 2023. Unlocking context constraints of LLMs: Enhancing context efficiency of LLMs with self-information-based content filtering. arXiv preprint arXiv:2304.12102.

Ling, C., Zhao, X., Lu, J., Deng, C., Zheng, C., Wang, J., Chowdhury, T., Li, Y., Cui, H., Zhao, T., et al., 2023. Beyond one-model-fits-all: A survey of domain specialization for large language models. arXiv preprint arXiv:2305.18703.

Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., He, H., Li, A., He, M., Liu, Z., et al., 2023. Summary of chatgpt/gpt-4 research and perspective towards the future of large language models. arXiv preprint arXiv:2304.01852.

Luo, Z., Xie, Q., Ananiadou, S., 2023. Chatgpt as a factual inconsistency evaluator for abstractive text summarization. arXiv preprint arXiv:2303.15621.

Ma, C., Wu, Z., Wang, J., Xu, S., Wei, Y., Liu, Z., Guo, L., Cai, X., Zhang, S., Zhang, T., et al., 2023. ImpressionGPT: An iterative optimizing framework for radiology report summarization with chatGPT. arXiv preprint arXiv:2304.08448.

Miller, D., 2019. Leveraging BERT for extractive text summarization on lectures. arXiv preprint arXiv:1906.04165.

OpenAI-Policy, 2023. OpenAI API data usage policy. https://openai.com/policies/api-data-usage-policies. (Accessed: 12 July 2023).

Pinecone, 2023. Vector database. https://www.pinecone.io/learn/vector-database/. (Accessed: 20 June 2023).

Reimers, N., Gurevych, I., 2019. Sentence-bert: Sentence embeddings using Siamese Bert-networks. arXiv preprint arXiv:1908.10084.

Schlag, I., Sukhbaatar, S., Celikyilmaz, A., Yih, W.-t., Weston, J., Schmidhuber, J., Li, X., 2023. Large language model programs. arXiv preprint arXiv:2305.05364.

Tan, Y., Min, D., Li, Y., Li, W., Hu, N., Chen, Y., Qi, G., 2023. Evaluation of ChatGPT as a question answering system for answering complex questions. arXiv preprint arXiv:2303.07992.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al., 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al., 2022. Chain-of-thought prompting elicits reasoning in large language models. Adv. Neural Inf. Process. Syst. 35, 24824–24837.

Wu, J., Antonova, R., Kan, A., Lepert, M., Zeng, A., Song, S., Bohg, J., Rusinkiewicz, S., Funkhouser, T., 2023. Tidybot: Personalized robot assistance with large language models. arXiv preprint arXiv:2305.05658.

Yang, X., Li, Y., Zhang, X., Chen, H., Cheng, W., 2023. Exploring the limits of chatgpt for query or aspect-based text summarization. arXiv preprint arXiv:2302.08081.

Zhang, H., Liu, X., Zhang, J., 2023a. Extractive summarization via chatgpt for faithful summary generation. arXiv preprint arXiv:2304.04193.

Zhang, H., Liu, X., Zhang, J., 2023b. SummIt: Iterative text summarization via ChatGPT. arXiv preprint arXiv:2305.14835.