Optimizing LLM API usage costs with novel query-aware reduction of relevant enterprise data

AREFEEN Md Adnan, DEBNATH Biplob, CHAKRADHAR Srimat

Abstract

Costs of LLM API usage rise rapidly when proprietary enterprise data is used as context for user queries to generate more accurate responses from LLMs. To reduce costs, we propose LeanContext, which generates query-aware, compact and AI model-friendly summaries of relevant enterprise data context. This is unlike traditional summarizers that produce query-unaware human-friendly summaries that are also not as compact. We first use retrieval augmented generation (RAG) to generate a query-aware enterprise data context, which includes key, query-relevant enterprise data. Then, we use reinforcement learning to further reduce the context while ensuring that a prompt consisting of the user query and the *reduced context* elicits an LLM response that is just as accurate as the LLM response to a prompt that uses the original enterprise data context. Our reduced context is not only query-dependent, but it is also variable-sized. Our experimental results demonstrate that LeanContext (a) reduces costs of LLM API usage by 37% to 68% (compared to RAG), while maintaining the accuracy of the LLM response, and (b) improves accuracy of responses by 26% to 38% when state-of-the-art summarizers reduce RAG context.

Keywords
generative AI, large language model, text summarization, natural language processing, reinforcement learning, domain-specific question-answering, retrieval augmented generation

1. Introduction

Large language models (LLMs) are advanced AI models trained on extensive textual data to generate human-like language, significantly enhancing natural language processing tasks. Notable examples, like OpenAI's GPT-4[1] feature user-friendly application programming interfaces (APIs), driving their widespread use in context-aware chatbots, real-time language translation, and efficient text summarization. This has led to enhanced user experiences across diverse industries.

LLMs like GPT-4 cannot answer queries about information in proprietary enterprise data because the LLM was not trained on this data. However, when LLMs are made aware of proprietary enterprise data, they can generate responses that use industry-specific jargon, processes, and context. This results in more accurate and relevant responses for enterprises.

Fine-tuning and Retrieval-Augmented Generation (RAG)[2] are two prominent methods employed to make LLMs aware of enterprise data. Fine-tuning changes the model weights of an LLM to adapt the model to domain-specific nuances. In contrast, RAG leverages pre-trained model (without any modifications) in conjunction with a retriever that selects relevant information (*context*) from enterprise data and incorporates this external knowledge in prompts to LLMs. In this paper, we use a RAG approach to make LLMs aware of enterprise data.

The cost of LLM API usage can add up very quickly, especially when incorporating enterprise information in prompts to LLMs. Cost depends on the number of tokens in the prompt and the LLM response. In GPT-3 LLM, a token is approximately 4 characters[3] but this varies across LLMs and languages.

As an illustration of the high costs of LLM API usage, consider a service with 15,000 visitors where every visitor sends 3 requests twice a week. A (representative) prompt has about 1800 prompt tokens and 80 output tokens[4]. Cost of GPT-4 API usage for a month works out to $21,200 (pricing of $0.03/1K tokens for the prompt, and $0.06/1K tokens for the generated output).

In this article, we focus on reducing the costs of LLM API usage in scenarios where use of enterprise data generates more useful responses. We propose *LeanContext*, a novel cost-efficient, query-aware enterprise data context retrieval system. The retrieved context is

compact, and highly relevant to answer the query. Our experimental results show that LeanContext (a) reduces the cost of LLM API usage (by 37% to 68% compared to RAG context), while maintaining high accuracy of responses, and (b) improves accuracy of responses (by 26% to 38%) when notable summarizers reduce RAG context.

2. Retrieval augmented generation

Fig. 1 shows the traditional retrieval augmented generation method. It consists of two distinct parts, that can operate in parallel: enterprise data ingestion, and query-response.

Enterprise data ingestion: Enterprise text documents are split into small *chunks* by a text splitter (a chunk is a set of consecutive sentences in an enterprise document). An embedding generator embeds each chunk in an *n*-dimensional vector, where *n* is pre-determined. These vectors, and the corresponding chunks, are stored in a vector database. Storing the chunks as vectors makes it easy to find enterprise data that is relevant to a given user query.

Query-response: A user query is also embedded in an *n*-dimensional vector. As shown in Fig. 1, a semantic search method determines the *N* vectors in the vector database that are most like the user query vector. Here, *N* is a pre-determined parameter. The chunks that correspond to these N vectors are the relevant *RAG context*. This context is combined with the user query to construct the prompt and the response from the LLM is provided to the user. The size of the prompt cannot exceed the max-token limit of the LLM API (this limit can vary across different LLMs).

3. LeanContext

We make two critical observations that are key to the design of LeanContext. First, experiments on real-world applications show that not all information in the *N* chunks of context retrieved by a traditional RAG is necessary for an LLM to generate an accurate response. Second, the specific information in the RAG context that can be omitted depends on *the user query*. LeanContext leverages these insights to construct a more compact *reduced context* from the RAG context. The reduced context directly leads to lower LLM API usage costs. Also, the reduced context results in an LLM response that is just as accurate as the LLM response for a prompt that uses the larger (*N* chunks of) RAG context.

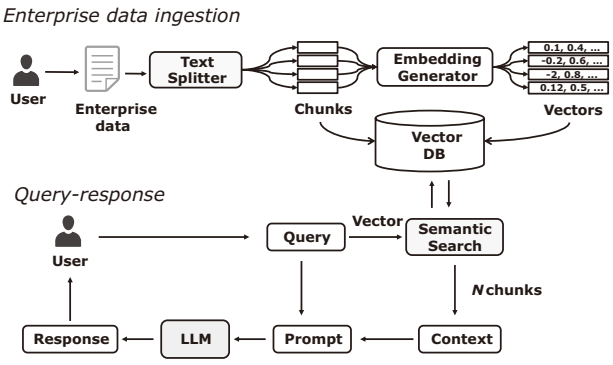

Fig. 1 Retrieval augmented generation.

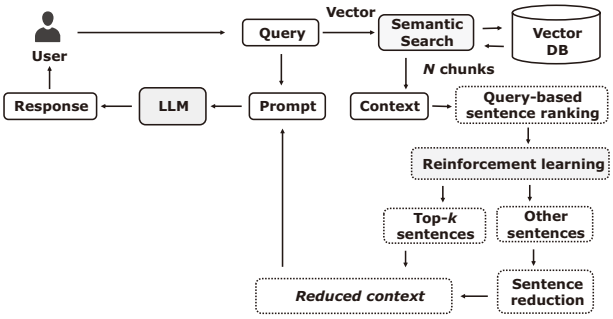

Fig. 2 System overview of LeanContext.

Fig. 2 shows the system overview of LeanContext. We use the traditional RAG method to retrieve the *N* chunks of enterprise data context. Then, we rank the sentences in this RAG context based on their relevance to the user query. A novel reinforcement learning algorithm (described in section 3.1) determines the top-*k* sentences in the ranked RAG context that should be considered to construct the reduced context. Our reinforcement learning algorithm determines the value of *k* based on the user query and the RAG context. Then, the important top-*k* sentences are left intact but the rest of the less important sentences in the ranked RAG context are further compressed using phrase deletion or summarization. The top-*k* sentences and the compressed less important sentences are now combined to create the *reduced context*, as described in section 3.2.

3.1 Reinforcement learning to compute *k*

Given a user query, and the corresponding RAG context, our novel lightweight Q-learning-based reinforcement learning (RL) algorithm computes a good *k* for this pair. We briefly describe the state, action and reward components of our RL algorithm.

State: We create an embedding vector for the RAG context of *N* chunks. Then, we derive a *difference vector* by

subtracting the query embedding vector (v_q) from the RAG context embedding vector (v_C). We construct difference vectors for many query-context pairs and cluster these vectors to compute centroids (we use K-Means clustering algorithm). These centroids are our state vectors $\mathcal{S}$.

$$\mathcal{S} \leftarrow \boldsymbol{K\text{-}Means}\left(\bigcup_{i,j}\left(v_{c_i} - v_{q_j}\right)\right)$$

The variables *i* and *j* are used to index the different RAG contexts and user queries, respectively.

Action: An action corresponds to a specific fraction of the total sentences in the ranked RAG context. An action can be any value from 0 to 0.4, each spaced 0.05 apart. For example, if an action assumes the maximum value of 0.4, then 40% of the total number of sentences in the ranked RAG context will be considered as top-*k* sentences that are most similar to the query. The *k* in top-*k* is derived as the product of the current action value and the total number of sentences in the ranked RAG context.

Reward: Given a value for action, we can determine the top-*k* sentences and their token count. We define token ratio (τ) as the ratio of the token counts of top-*k* sentences in reduced context and the all the sentences in the ranked RAG context. The lower the token ratio, the smaller the top-*k* context length. However, the accuracy of the LLM response for the top-*k* reduced context must be comparable to the accuracy of LLM response for the full RAG context. We use ROUGE-1[5] scores to compare the accuracies of different LLM responses (the reference response we use to compute the ROUGE-1 scores is described in section 3.3). If the ROUGE score with the full RAG context is (r^*) , and the ROUGE score for top-*k* context is *r*, then the current (state, action) pair value in Q table will be rewarded if $r - r^* \geq 0$, otherwise it will be penalized. The reward function $\boldsymbol{R}$ is defined as follows.

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

Here, α controls the relative contribution of the token ratio and accuracy of the response to the reward value.

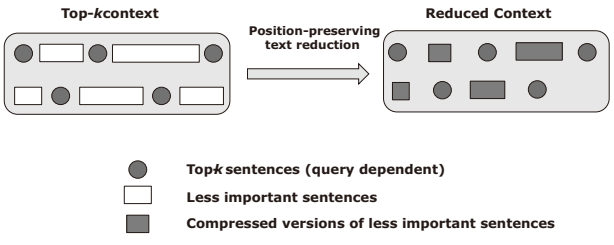

Fig. 3 Construction of Reduced Context.

3.2 Reduced Context

The RL algorithm determines the query-specific value of *k*, which determines the top-*k* context. This context includes the important top-*k* sentences that are related to the query, as well as other less important sentences around the top-*k* sentences. **Fig. 3** shows how we construct the Reduced Context. We leave the most relevant top-*k* sentences intact because they are critical for maintaining the relevance of the context to the query. However, the less important sentences are individually compressed further using open-source text reduction methods[6)-10)]. We also do not include in the reduced context any sentences that are beyond the last top-*k* sentence.

We preserve the original order of both the top-*k* sentences and the less important sentences in the ranked RAG context. By preserving the sentence order, we ensure the temporal coherence of the context. Such a holistic approach of constructing the reduced context ultimately results in preserving the accuracy of LLM responses, while significantly reducing the cost of LLM API usage.

3.3 Results

Enterprise data: We use arXiv and BBC-News data repositories. They include documents published in March 2023[6] which were not used to pre-train GPT-3.5-Turbo model. We randomly chose 25 documents from arXiv. They have 63 to 962 sentences, and 352 sentences per document on average. Similarly, we chose 100 documents from BBC News. They have 4 to 139 sentences, with 30 sentences per document on average.

Queries and reference responses: We generated 100 queries for each dataset by using QAGenerationChain[11],

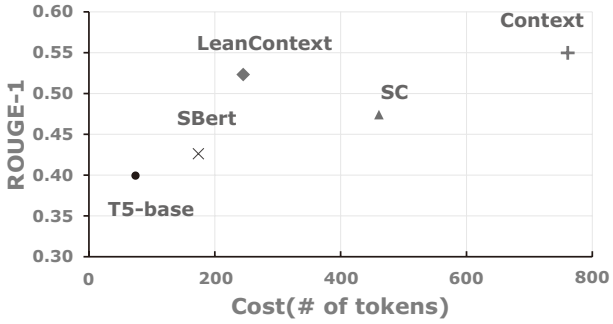

Fig. 4 Comparison of LeanContext with methods where notable summarizers reduce RAG context.

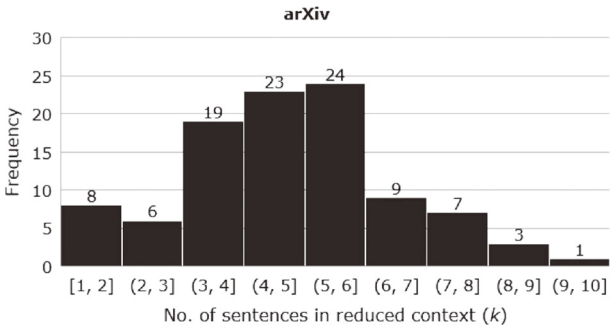

Fig. 5 Distribution of (important sentences) *k* for arXiv data.

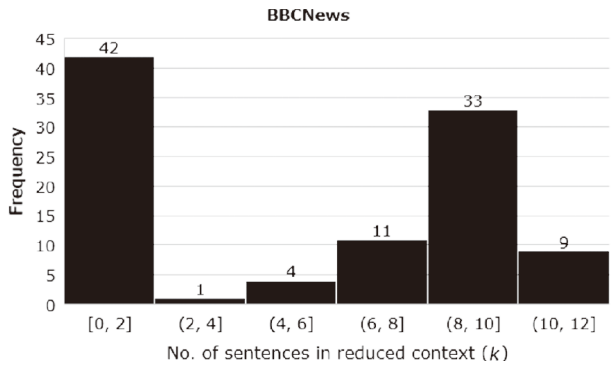

Fig. 6 Distribution of (important sentences) *k* for BBC News data.

which uses full documents as context to get LLM responses that are the reference for computing ROUGE-1 accuracy scores for RAG or LeanContext.

Enterprise data context: We set *N=4 and N=8* for the arXiv and BBC News, respectively. 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 the BBC News data, the total number of sentences in the context varied from 18 to 34, with an average of 26 sentences per context (**Fig. 4**).

Reduced context: For the arXiv and BBC News data, the distribution of top-*k* sentences are shown in **Fig. 5** and **Fig. 6**, respectively.

Table 1 compares the impact of RAG context and Reduced context. The accuracy (ROUGE-1 score) of LLM responses is similar, but the Reduced context lowers LLM API usage costs by 37% to 68%.

Compared to other notable text reduction models like T5[10], BERT[9], and SC[6], LeanContext reduces cost and improves accuracy (Fig. 4). It also boosts the accuracy (ROUGE-1 score) of responses when other notable summarizers are used to reduce the RAG context (**Table 2**).

Table 1 Comparison of our Reduced context with traditional RAG (Accuracy is ROUGE-1 score).

Data	Context	Accuracy	Tokens (Prompt, Response)	Cost Savings
arXiv	RAG (*N*=4)	39.85	521, 26	-
	Reduced	38.44	321, 22	37%
BBC	RAG (*N*=8)	54.98	724, 37	-
	Reduced	52.33	218, 27	68%

Table 2 Improving accuracy of responses when other notable summarizers reduce RAG context.

Data	Summarizer	ROUGE-1	Improvement
BBC-News	SBert	42.61	-
	SBert+ LeanContext	53.55	26%
	T5	39.93	-
	T5 + LeanContext	55.32	38%

4. Conclusion

LeanContext is a cost-efficient query-aware context reduction system to mitigate the cost associated with LLM API usage, while maintaining high accuracy. Reduction in context also improves the inference time of LLMs. LeanContext can also be used effectively in conjunction with other notable summarizers to reduce RAG context.

References

1) OpenAI: GPT-4
https://openai.com/product
2) Patrick Lewis et al.: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, 34th Conference on Neural Information Processing Systems (NeurIPS), 2020
https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf
3) Claudia Slowik: How Much Does It Cost to Use GPT Models? GPT-3 Pricing Explained, Neoteric, 2023.2
https://neoteric.eu/blog/how-much-does-it-cost-to-use-gpt-models-gpt-3-pricing-explained/
4) Lingjiao Chen, Matei Zaharia and James Zou: FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance, 2023
https://arxiv.org/abs/2305.05176
5) Chin-Yew Lin: ROUGE: A Package for Automatic Evaluation of summaries, The Workshop on Text Summarization Branches Out (WAS), pp.74-81, 2004
https://aclanthology.org/W04-1013/
6) Yucheng Li: Unlocking Context Constraints of LLMs: Enhancing Context Efficiency of LLMs with Self-Information-Based Content Filtering, 2023
https://arxiv.org/abs/2304.12102
7) Md Tahmid Rahman Laskar, Mizanur Rahman, Israt Jahan, Enamul Hoque and Jimmy Huang: CQSumDP: A ChatGPT-Annotated Resource for Query-Focused Abstractive Summarization Based on Debatepedia, 2023
https://arxiv.org/abs/2305.06147
8) Henry Gilbert, Michael Sandborn, Douglas C. Schmidt, Jesse Spencer-Smith and Jules White: Semantic Compression With Large Language Models, 2023
https://arxiv.org/abs/2304.12512
9) Nils Reimers and Iryna Gurevych: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp.3982-3992, 2019
https://aclanthology.org/D19-1410/
10) Colin Raffel et al.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, Journal of Machine Learning Research volume21, 2020
https://jmlr.org/papers/volume21/20-074/20-074.pdf
11) GitHub: LangChain
https://github.com/hwchase

Authors' Profiles

AREFEEN Md Adnan
Research Assistant
NEC Laboratories America
University of Missouri-Kansas City

DEBNATH Biplob
Senior Researcher
NEC Laboratories America

CHAKRADHAR Srimat
Department Head
NEC Laboratories America

Information about the NEC Technical Journal

Thank you for reading the paper.
If you are interested in the NEC Technical Journal, you can also read other papers on our website.

Link to NEC Technical Journal website

Japanese

English

Vol.17 No.2 Special Issue on Revolutionizing Business Practices with Generative AI
– Advancing the Societal Adoption of AI with the Support of Generative AI Technologies

Remarks for Special Issue on Revolutionizing Business Practices with Generative AI
Approaches to Generative AI Technology: From Foundational Technologies to Application Development and Guideline Creation

Vol.17 No.2
June 2024

Special Issue TOP

Papers for Special Issue

Market Application of Rapidly Spreading Generative AI
NEC Innovation Day 2023: NEC's Generative AI Initiatives
Streamlining Doctors' Work by Assisting with Medical Recording and Documentation
Using Video Recognition AI x LLM to Automate the Creation of Reports
Understanding of Behaviors in Real World through Video Analysis and Generative AI
Automated Generation of Cyber Threat Intelligence
NEC Generative AI Service (NGS) Promoting Internal Use of Generative AI
Utilization of Generative AI for Software and System Development
LLMs and MI Bring Innovation to Material Development Platforms
Disaster Damage Assessment Using LLMs and Image Analysis

Fundamental Technologies that Enhance the Potential of Generative AI
NEC's LLM with Superior Japanese Language Proficiency
NEC's AI Supercomputer: One of the Largest in Japan to Support Generative AI
Towards Safer Large Language Models (LLMs)
Federated Learning Technology that Enables Collaboration While Keeping Data Confidential and its Applicability to LLMs
Large Language Models (LLMs) Enable Few-Shot Clustering
Knowledge-enhanced Prompt Learning for Open-domain Commonsense Reasoning
Foundational Vision-LLM for AI Linkage and Orchestration
Optimizing LLM API usage costs with novel query-aware reduction of relevant enterprise data

For AI Technology to Penetrate Society
Movements in AI Standardization and Rule Making and NEC Initiatives
NEC's Initiatives on AI Governance toward Respecting Human Rights
Case Study of Human Resources Development for AI Risk Management Using RCModel

NEC Information

2023 C&C Prize Ceremony