ICIC Express Letters
Volume **14**, Number **5**, May **2020**

ICIC International ⓒ2020 ISSN 1881-803X
pp. **1–8**

Comparing the performance of question answering by a large-scale language model on a resource-constrained machine

Ponrudee Netisopakul and Sira Haruethaipree

School of Information Technology, King Mongkut's Institute of Technology Ladkrabang
1 Chalong Krung, 1 Alley, Lat Krabang, Bangkok 10520
64607065@kmitl.ac.th

Received October XXX; accepted January XXX

ABSTRACT. *The development of large language models (LLMs) like ChatGPT and Google Bard has led to the creation of intelligent chatbots and question-answering systems that are gaining widespread popularity. However, there are still limitations in using LLMs to develop applications, including the substantial computational resources required for finetuning and deployment. This paper studies and experiments with two techniques to reduce the computing resources required for developing a question-answering system using LLMs. A quantization technique is employed to compress the model's size, and the application of Retrieval-Augmented Generation (RAG) techniques is utilized for information retrieval. The study compares the performance of compressed-size models using Quantization and RAG against the original-sized models. The results show that quantizing the model can compress the VRAM resources used in the GPU between 38% to 57% while still achieving 68.9% accuracies comparing to 70% in the non-compress model.*

Keywords: Large language models, Document Question Answering (DQA), Retrieval-Augmented Generation (RAG), Quantization, Resource-constrained machine.

1. Introduction.

The growth of artificial intelligence in natural language processing (NLP) is of great interest today, as there exist large language models (LLMs) with billions of parameters. These LLMs have been applied to the creation of chatbots, leading to the widespread adoption of applications such as ChatGPT [1] and Google Bard [2]. These applications can interact with users, answer questions, and provide guidance on problem-solving. Although chatbots powered by artificial intelligence, such as ChatGPT and Google Bard, can answer a wide range of questions for users, they still have limitations. For example, they cannot answer questions in some specific domains, or answer questions about recent events. This makes it necessary for organizations to finetuning the pre-trained model with a specific set of document data. However, the finetuning process and the generation of answers from a large language model require a relatively high amount of processing resources. This paper studies two techniques, namely *Quantization* and *Retrieval Augmented Generation (RAG)*, helping to utilize LLMs based document question-answering system on a machine with limited resources.

Quantization [3] is a technique for reducing the size of a model on the memory, making it easier to deploy large language models. This is done by converting the number of bits in the model from float32 or float16 to INT8 or INT4. Retrieval Augmented Generation (RAG) [4] is a technique that combines retrieval and generation techniques applied to document question-answering applications without finetuning.

This study conducted experiments on a computer with limited resources setting, compared model performance and memory usage on various sizes of Quantization. The research questions are:

1. Can a large language model be applied to a document question-answering system using only a computer with limited resources.

2. Can we use the principles of Retrieval Augmented Generation for creating a document question-answering system without finetuning LLMs.

3. What are the comparative performances of a LLMs-based document question-answering system when using various sizes of quantization.

The scope of this study is as follows. (1) The large language model used is an open-source model called Llama2-7b-chat [5], which has 7 billion parameters. (2) The method for the model compression technique used is called Absolute maximum quantization. Three sizes of quantization will be compared in the experiment; those are FP16, INT8, and INT4. (3) The document question-answering system will operate through the Langchain framework.

The organization of this paper is as follows. Section 2 reviews related works and technologies. Section 3 presents research methodology, including dataset preparation, DQA construction by applying RAG technique, and experimental design. Section 4 presents the experimental results and discussion. Finally, section 5 provides the conclusions and suggestions for future research.

2. Related works and technologies

2.1 Quantization. Large language models, which contain weights, vectors, and input sequences, are stored in random access memory (RAM). Currently, most large language models have billions of parameters and are stored in FP32 and FP16 formats. Loading the weights of a 1 billion parameter model requires approximately 4 GB of RAM in FP32 format, while only 2 GB of RAM is required in FP16 format. For example, running the Llama-2-7b-chat prototype model with FP32 bits requires approximately 28 GB of memory, which is much more than the hardware resources most people can support. Therefore, this study will start by comparing the performance of models with FP16 size and lower.

Absolute maximum quantization [6] is a method of reducing the number of bits in a model. It can be used to compress the size of a model to 8-bits and 4-bits by adjusting the weights. For example, equation (1) shows how to compress the number of bits from FP16 weights to INT8 weights, which has a total of 8 bits and can store 255 different values in the range of [-127,127]. The reduction of the model size to INT4 format is similar to the INT8 quantization method, but it only has 4 bits, which can store 15 different values in the range [-7,7].

$$X_{quant} = round \frac{127}{\max|X|} \cdot X \tag{1}$$

$$X_{dequant} = \frac{max|x|}{127} \cdot X_{quant} \tag{2}$$

Equation 1 is the quantization equation. It divides the maximum value of INT8, which is 127, by max|X|, which represents the maximum value of weights within the input dataset. This number is called the scaling factor, as it will be used to adjust each input weight in the input dataset or layer by multiplying to each input vector X , then rounded the value to an integer. Equation 2 is the de-quantization equation, which converts the weight value from INT8 to FP16. As can be seen in Figure 1, the converted weight value will be slightly different, which means that the conversion of the model size may affect the model's accuracy.

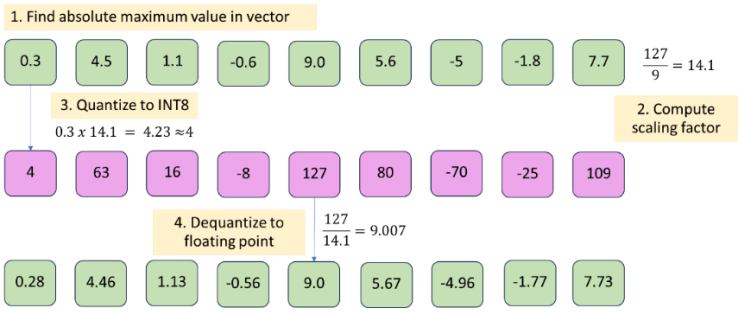

Figure 1. Shows an example of Absolute maximum quantization process using INT8 format [7] .

When the model is called to generate an answer, the model will load those INT4 or INT8 weight values and convert them back to FP16 vector format for calculation with the input vector. This method is called de-quantize as shown in equation 3.

$$Y = X * dequantize(W) \quad (3)$$

After quantization, the model is converted to GGML format [8] GGML is a library written in C that makes it easy and flexible to use machine learning models. It supports a variety of quantization formats, including 4-bit, 5-bit, and 8-bit. Models stored in GGML format can be loaded and deployed on both CPU and GPU through the llama.cpp library [9]

2.2 RAG (Retrieval Augmented Generation) was invented in 2020 to solve the problem of pre-trained language models used for text generation or chatbots. Instead of fine-tuning with a specific document dataset to answer questions for specialized tasks. RAG uses the Retriever method to help find the closest text in the document to the question and then sends the closest result to the large language model to generate the output, eliminating the need for fine-tuning.

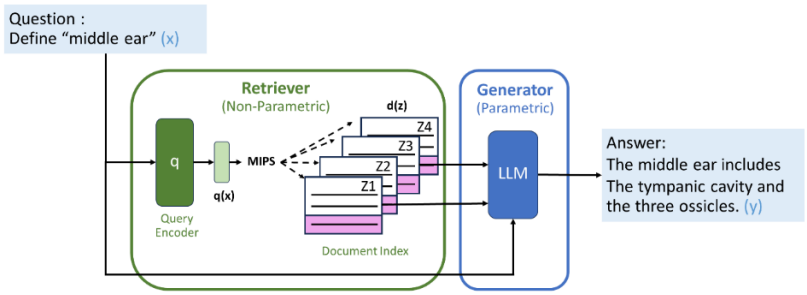

Figure 2. RAG combines the Retriever method with a pre-trained model [4]

Figure 2 shows the overview of RAG architecture, which combines the Retriever method with a pre-trained model. Here, x is the query that will be sent to match the document set to find the Maximum Inner Product Search (MIPS) according to the specified top-k number, which is replaced by z, while y is the final output. The architecture consists of two main parts:

1. Non-Parametric memory component: Instead of fine-tuning the model as usual, it is replaced with a relevance retrieval method. The query x, which has been converted to a vector format, is used to match the vectors of the document set to obtain a similarity score. In Figure 2, Maximum Inner Product Search (MIPS) is replaced with the Euclidean distance (L2) method. The documents with the highest similarity scores to the query are then used in the parametric section. The number of documents depends on the specified k value.
2. Parametric memory component: This component uses a pre-trained text generation model to generate answers from the answers obtained from the non-parametric

component. Usually, it utilizes the transformer architecture, which has the decoder part or both the encoder and decoder, such as BART, T5, etc.

To find the documents that are most likely to answer the question, it is necessary to find the similarity using the Euclidean distance method (L2) as shown in equation 4.

$$i = k - argmin_i ||x - y_i|| \quad (4)$$

From equation 4, the symbol ||.|| means Euclidean distance (L2), x is the query or the question, which is compared to the entire text document represented by y to find the most similar piece of text. Both x and y are represented in the form of vectors. The smaller the output value, the more similar the document is to the question. The selection of the results is similar to the principle of the nearest neighbor (Nearest neighbor) algorithm, that is, selecting the top-k most similar documents y to answer the question x.

2.3 Large language models (LLMs) are language models based on the transformer architecture with hundreds of millions to billions of parameters or more and are trained on large datasets. Examples of LLMs are GPT-3 [10] LLaMa [11] etc. In this study, we use a language model called Llama2-7b-chat[5], which was developed by Meta. It has 7 billion parameters, can support 4096 tokens in one input, and is fine-tuned on a conversational dataset to make the model good at answering questions in a chatbot-like interactive format. Reinforcement Learning with Human Feedback (RLHF)[12] is used as a supplement to make the model answer questions in a way that is close to what humans expect.

2.4 Text embedding models are models that represent text in the form of vectors. It is used to find the most similar meanings of text within the vector space. Previous embedding model, such as the Glove model [13] has insufficient ability to learn from surrounding context; while the transformer architecture has improved the ability to learn from the context of words through the self-attention method. BERT [14] text embedding model, which is a pre-trained model with a large dataset, can be used to create a text embedding model. Later, SBERT [15] showed that the benefits of fine-tuning through the transformer architecture model can improve the performance of text embedding models. This led to competition for the performance of text embedding models through model tuning techniques. Examples of text embedding models include text-embedding-ada-002 [16] sentence-t5 [17] etc.

Technologies and tools used in this research are as follows. Langchain [18] is a framework to create applications with the RAG technique, from data retrieval to large language models (LLMs) response generation. First, Llama-2 in GGML format is downloaded from HuggingFace [19] then, the designed experiments are conducted on Kaggle [20] to see memory usages and accuracies in different quantized formats, which are FP16, INT8, and INT4. Faiss [21] library is used to calculate text vector similarities during the RAG retrieval step. Finally, HuggingFace space [19] is used to deploy and demonstrate the question-answering application.

3. Methodology

3.1 Dataset. The data used in this study to compare models is the Wikipedia subset of the TriviaQA dataset [22] This subset is a reading comprehension dataset that contains about 77,000 rows of data collected from Wikipedia articles. Each row of data contains question-answer-evidence triples. The last evidence text related to the question and answer will be used with the RAG method. Due to the huge time-consuming nature of the experiments, the results presented in this paper are obtained only first 1000 rows of the validation subset, which contains 7993 rows, comprising of 12.5% of the dataset.

3.2 Document Question Answering System Construction

Figure 3 shows the overview steps of creating a question-answering system using the

technique called RAG (Retrieval Augmented Generation) through the Langchain tool. In short, there are four steps: data preparation, text-to-vector conversion and storage, similar text retrieval from the document using RAG, and finally, answer generation.

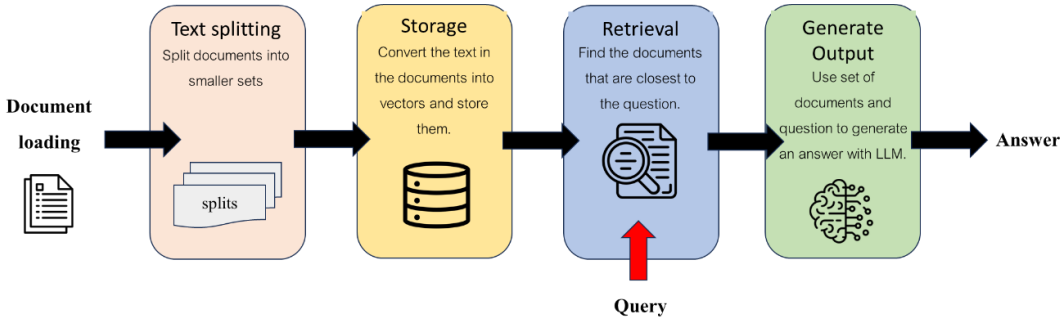

Figure 3. a DQA system construction process using the RAG method and the Langchain library.

3.2.1 Data Preparation. In general, document datasets are usually imported in the form of text, with varying lengths. When used with natural language models, this can cause errors because each model has a maximum supported input token size. In particular, the Llama-2-7b-chat model used has a maximum token size of 4096. Therefore, all input text must go through a document splitting process to fit the model.

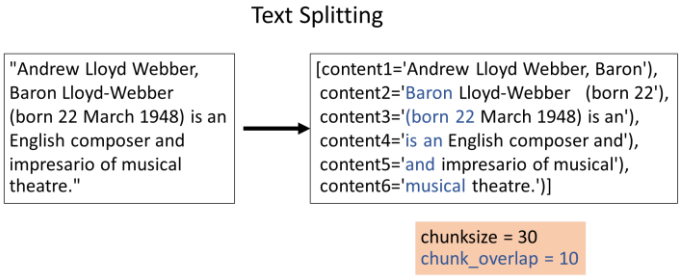

Figure 4. Example of the document splitting process.

From Figure 4, the document will be divided into chunks, each limited to 30 characters, and the blue characters indicate the overlapping parts specified by the chunk overlap parameter.

Chunk size is the number of characters that will be stored in a chunk. Chunk Overlap specifies how many characters of the previous text will overlap. This study sets chunk size to 1000 and chunk overlap to 200.

3.2.2. Text to vector conversion and storage. Word vectors of chunked text are created using an embedding model. These vectors are stored to be used in the next retrieval step. We chose the Thenlper/gte-base [23] embedding model because the model is smaller than other models in the top ten list of the Massive Text Embedding Benchmark (MTEB) [24] ranking. Faiss is used as a vector store for the converted text vectors.

3.2.3 Retrieval-Augmented Generation (RAG). RAG is the Non-Parametric component of the system. It takes chunked text vectors stored in FAISS and compares to the query vector. The similarity scores are calculated and ranked using the Euclidean distance. Chucks with Top-K similarity will be selected for generating the answer in the next step. In the document question-answering system, we expect that the matching answer from the document should not be in many places. In addition, the larger K will result in longer computing time. Therefore, in this study, we start by setting K equals to 3.

3.2.4 Prompt and Generate. Prompt is an instruction to the language model to guide its response. Prompt can help the language model to understand the context and generate

responses that are consistent with the task. Prompt in this study consists of three parts as shown in Figure 5. The command part specifies the context of the question and the response's format. For example, the command part may specify that the model should generate an answer of at least three sentences, or that the model should answer "I don't know" if it is not sure of the answer. The context part is the top-k text from the previous retrieval step. The question part is the question submitted to the RAG method. Both context part and question part will be replaced with a ranked response and a question from TriviaQA.

custom_prompt_template = "**Use the following pieces of context to answer the question at the end. If you don't know the answer, just say that you don't know, Don't try to make up an answer. Please generate answer not too long and easy understand.**"
Context : {**context**}
Question : {**question**}
Generated answer …

Figure 5. Example prompt uses to generate an answer.

3.3 Experimental Design. There are two sets of experiments comparing the quantized language models performance with the original model. The first experiment measures the amount of memory usage when varying the number of input tokens. The second experiment measures accuracies of the models compared to solutions from the TriviaQA dataset.

3.3.1 Memory Usages with Varied Input Tokens. The Llama2-7b-chat model used in the experiment can support an input length of 4096 tokens. This means that the model can understand long text. However, the longer the input, the more computational resources are required to generate a response. The experiment feeds inputs with token lengths of 1000, 2000, 3000, and 4000 to models of different quantized sizes: FP16, INT8, and INT4. These experiments are conducted on 10 samples from the TriviaQA dataset to measure the average memory usage in megabytes (MB).

3.3.2 Question Answering Performance of Different Quantized Models. The accuracy of question answering for each quantized model (FP16, INT8, and INT4) is compared using the TriviaQA dataset of 1000 rows. The accuracy of each model is measured and compared between the original pre-trained models and the RAG technique, both without fine-tuning.

The accuracy is measured using the Exact Match percentage *(EM)* formula shown in equation 5 [25] Where *M* represents the number of questions that the model answered correctly, and *N* represents the total number of questions in the experiment.

$$EM = \frac{M}{N} X\ 100 \tag{5}$$

For example, if there are a total of 100 questions, the model can answer 50 questions correctly. This makes the *EM* accuracy score equal to 0.5 or 50%.

4. Results and Discussion. The experimental results comparing the performance of the quantized models are divided into two parts. The first part discussed the experimental results for the memory usage for each quantized model when feeding inputs with different numbers of tokens. The second part discussed the experimental results of the question-answering system performance of each quantized model.

4.1 Comparison Results Of Memory Usages with Varied Input Tokens. From the experimental design in section 3.3.1, the llama2-7b-chat model was quantized, from FP16

size to the models with INT8 and INT4 sizes. Each was tested to measure the amount of memory usage for generating answers with varied sizes of input tokens 100, 1000, 2000, 3000, and 4000, respectively.

The results shown in Figure 6 show that the Llama2-7b-chat model, when feeding the input length of 4000 tokens with the model size of FP16, used up to 15,842 MB or 15.8 GB of memory. This suggests that at least 16 GB RAM should be allocated on a PC for one to run the model in FP16 format. Meanwhile, the quantized model size of INT8 used only 9,829 MB or 9.8 GB of memory, reducing FP16 by 6,013 MB or 6.0 GB. While the quantized INT4 model can compress the amount of memory used in calculations to only 6,855 MB or 6.8 GB. The results show that quantizing the model can compress the VRAM resources used in the GPU between 38% in the INT8 model and 57% in the INT4 model.

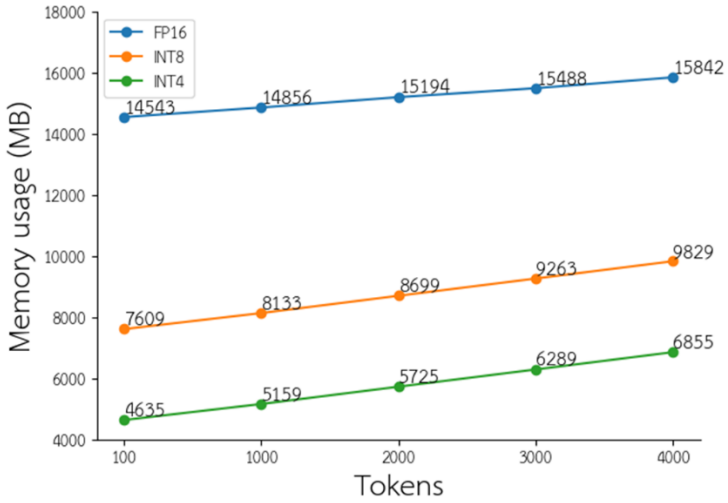

Figure 6. Show the amount of memory used for each size of the model, compared to the number of token input sizes.

4.2 Comparison Results of Question Answering Performance From the experimental design in section 3.3.2, the experiment compares question-answering performance across different quantized models in both pre-trained model and RAG-based approaches.

Table 1. The accuracy of each quantized model measured in Exact Match percentage, comparing a pretrained model to pretrained with RAG technique (zero-shot learning)

Model type	Pretrained (%)	Pretrained + RAG (%)
FP16	59.0	**70.0**
INT8	**61.2**	68.2
INT4	60.6	68.9

Table 1 revealed the performance of the question-answering system for each size of pre-trained models (the second column) and pretrained with RAG technique (the third column), both without finetuning. The accuracy ranged from 59.0 - 61.2 %, for the pretrained models, and from 68.2-70.0% for the pretrained with RAG models. This showed improvements between 7%-11%. Upon the investigation of answers mismatched by the pretrained model versus answer matched correctly by pretrained with RAG model, answers containing proper names from specific domains are often found. Hence, the study suggests that when lacking a specific domain knowledge in the context, the pretrained model often generate “hallucinated” answers which is not the correct answer. Therefore,

the RAG technique not only helps to eliminate the finetune step but also helps to improve the DQA accuracies.

However, the experiment results in the same model (the same column) are about the same, regardless of the model sizes. The accuracies are between 59.0-61.2 for the pretrained models and between 68.2-70 for the pretrained with RAG models. Hence, reducing the bit number from FP16 to INT8 and INT4 did not significantly affect the performance of the models, either pretrained only or pretrained with RAG.

5. Conclusion. To implement a question-answering system using large language models on limited resource machines, this paper presented a quantization technique that can compress the amount of memory used, and a RAG technique that can be used with pretrained model without the need for fine-tuning, The two experiments showed that this solution can reduce the memory usages, save time and resources that may be unattainable for fine-tuning, while still keeping the performance of the question answering system. The trained model is deployed on HuggingFace's 8vCPU 32GB RAM for a demonstration purpose.

The method presented in this study is just one approach that aids in reducing memory usage. Meanwhile, there are various techniques for model quantization available today, such as NF4 [26] which employs quantization techniques to adjust weight values within the range of -1 to 1, following a uniform distribution. This helps address outliers and enhance model accuracy. Another example is GPTQ [27] which utilizes post-training quantization, requiring testing with a dataset to facilitate the comparison of loss values before and after adjustment, aiming for minimal loss. We hope that this study can provide a guideline for readers who need to employ large language models with a limited resource machine.

REFERENCES

[1] OpenAI, Introducing ChatGPT, https://openai.com/blog/chatgpt, Accessed 30 March 2022.

[2] S. Pichai, An important next step on our AI journey, Google the keyword, https://blog.google/technology/ai/bard-google-ai-search-updates/ , Accessed 2023.

[3] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney and K. Keutzer, A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv preprint arXiv: 2103.13630, 2021.

[4] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, & D. Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, vol. 33, pp. 9459-9474, 2020.

[5] Meta, Introducing Llama 2 The next generation of our open source large language model, https://ai.meta.com/llama/ , Accessed 2023.

[6] M. Labonne, Introduction to Weight Quantization, https://towardsdatascience.com/introduction-to weight-quantization-2494701b9c0c, Accessed 2023.

[7] Int8 about Machine learning, Local Large Language Models, https://int8.io/local-large-language-models-beginners-guide/, Accessed May 31, 2023.

[8] G. Gerganov, ggml, github, https://github.com/ggerganov/ggml. Accessed 2023.

[9] G. Gerganov, Llama.cpp, github, https://github.com/ggerganov/llama.cpp. Accessed 2023.

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, & D. Amodei, Language Models are Few-Shot Learners. Advances in neural information processing systems vol. 33, pp. 1877-1901, 2020.

[11] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P.

Bhargava, S. Bhosale, D. Bikel, Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv preprint arXiv:2307.09288, 2023.

[12] Z. Li, Z. Yang, & M. Wang, Reinforcement Learning with Human Feedback: Learning Dynamic Choices via Pessimism. arXiv preprint arXiv:2305.18438, 2023.

[13] R. Brochier, A. Guille, & J. Velcin, Global Vectors for Node Representations. The World Wide Web Conference, pp. 2587-2593, 2019.

[14] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of naacL-HLT. Vol. 1, pp. 2, 2019.

[15] N. Reimers, I. Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv preprint arXiv:1908.10084, 2019.

[16] R. Greene, T. Sanders, L. Weng, Arvind Neelakantan New and improved embedding model, OpenAI, https://openai.com/blog/new-and-improved-embedding-model, December 15, 2022, Accessed 2023.

[17] J. Ni, G. H. Ábrego, N. Constant, J. Ma, K. B. Hall, D. Cer, & Y. Yang, Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models. arXiv preprint arXiv:2108.08877, 2021.

[18] LangChain, Get your LLM application from prototype to production, https://www.langchain.com/, Accessed 2023.

[19] Hugging Face, The AI community building the future, https://huggingface.co/ , Accessed 2023.

[20] Kaggle: Your Machine Learning and Data Science Community, https://www.kaggle.com/ , Accessed 2023.

[21] Faiss, Welcome to Faiss Documentation, https://faiss.ai/index.html, Accessed 2023.

[22] M. Joshi, E. Choi, D. S. Weld, & L. Zettlemoyer, TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. arXiv preprint arXiv:1705.03551, 2017.

[23] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, & M. Zhang, Towards General Text Embeddings with Multi-stage Contrastive Learning. arXiv preprint arXiv:2308.03281, 2023.

[24] HuggingFace Spaces, Massive Text Embedding Benchmark (MTEB) Leaderboard, https://huggingface.co/spaces/mteb/-leaderboard , Accessed 2023.

[25] Bai, Y., & Wang, D. Z. More than reading comprehension: A survey on datasets and metrics of textual question answering. arXiv preprint arXiv:2109.12264, 2021.

[26] T. Dettmers, A. Pagnoni, A. Holtzman, & L. Zettlemoyer, QLoRA: Efficient Finetuning of Quantized LLMs. arXiv preprint arXiv:2305.14314, 2023.

[27] E. Frantar, S. Ashkboos, T. Hoefler, & D. Alistarh, GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. arXiv preprint arXiv:2210.17323, 2022.

Point-by-point Response to Reviewer's Comments

Reviewer 1		
Reviewer's comments	**Point-by-point response**	**Page number**
This paper proposed a method based on quantization technique to improve performance of question-answering system. Results of three models are compared, and improvements in terms of memory usage are shown. Generally, this paper is clearly explained. Some suggestions for improving quality of the paper are as follows: (1) In Figure 1 & 2, font size of characters are too small. It is difficult to read.	Thank you for your comments. We have modified figure 1 and 2 to ensure that all font sizes are readable.	Figure 1 and figure 2 – page 3
(2) There are a number of abbreviations, the full name of each term should be put at the first mention.	The authors have thoroughly checked all abbreviations and put the full terms with the abbreviation at the first mention before using the abbreviations in following mentions.	Abstract, Introduction and Related work – page 1-2

Reviewer 2		
Reviewer's comments	**Point-by-point response**	**Page number**
1. Did the author train the large language model on the Kaggle? However, the author presented in the abstract, "This paper demonstrates an approach to develop a question-answering system with a limited resource computer." Why does the author not train the language model on the PC?	It is possible that the abstract is misleading. We have corrected it as followed "*This paper studies and experiments with two techniques to reduce the computing resources required for developing a question-answering system using LLMs.*" The experiments presented in this paper (figure 6) were conducted on Kaggle with GPUs because the authors' PC has only 4 GB RAM (4000 MB), which is not viable to run the experiment. Based on figure 6, the authors would suggest that PCs should have at least 16 GB to run the FP16 model. The authors also add statements to clarify this point in section 2.4 and section 4.1.	Abstract – page 1 Section 2.4 -page 4 and section 4.1 – page 7
2. Did the author replicate Figures 1 and 2 or copy the original images (papers [8] and [4])? If the author copied the	Thank you for your suggestion. Figures 1 and 2 are redrawn based on your suggestion.	Figure 1 and figure 2 – page 3

Reviewer 2		
Reviewer's comments	**Point-by-point response**	**Page number**
original images, please replicate these Figures.		
3. Could the author remove Section 2.2, technologies and tools used? However, the author could briefly explain all these technologies and tools, may be present in one or two lines.	Section 2.2 are removed and replaced with a short paragraph explained all the technologies used in this study.	The last paragraph of Section 2 - Page 4
4. As shown in Table 1, why does the author highlight (bold fonts) at "61.2%"?	In table 1, 61.2% is the highest accuracies for the pretrained model, which are obtained from INT8 format; while 70% is the highest accuracies for the pretrained+RAG model. We did not change anything in table 1.	Table 1 – page 7
5. Could the author compare the proposed method with other research?	At the time of our study, we did not find any research that use the same setting as our study. There are other settings, such as in reference [26-27] which may not be fair to compare to ours.	The last paragraph in the Conclusion section – page 8

Reviewer 3		
Reviewer's comments	**Point-by-point response**	**Page number**
- The abstract should contain the results.	The authors added the results in the abstract as suggested.	Abstract-page 1
- Figure names of Figures 2 to 5 are too long.	We rewrite the captions of Figure 2 to Figure 5, moving the details explanation into the context.	Figure 1 - figure 5 's captions – Page 3,5,6
- In 3.2.2, The authors mention "Chucks with Top-K similarity", Why K = 3 please describe more detail.	In the document question-answering system, we expected that the matching answer from the document should not be in many places. In addition, the larger K will result in longer computing time. Therefore, in this study, we started by setting K=3. We add a sentence to explain this point in the paper.	Section 3.2.3 – page 5-6
- In 3.3.2 "The accuracy of question answering for each quantized model (FP16, INT8, and INT4) is compared using the TriviaQA dataset of 1000 rows". The authors should describe, why 1000 rows of data for testing?	The experiments are very time-consumming; processing 1000 rows of TriviaQA in DQA system took us 2 hours 33 mins for the INT4 model. Six experiments on 1000 rows took us about 15 hours. The authors think that experiment with 1000 rows of data (comprises 12.5% of of TriviaQA Wikipedia validation set, which contains 7993	Section 3.1 – page 4

Reviewer 3		
Reviewer's comments	**Point-by-point response**	**Page number**
	rows) is enough to proof the crucial point of proposed idea. The authors have clarified more about the dataset in the section 3.1.	
- The authors should describe Why the EM(Exact Match) metric was chosen for evaluating the performance of the question-answering models?	The authors use the Exact Match (EM) measurement based on this reference which is added into the manuscript. *Bai, Y., & Wang, D. Z. More than reading comprehension: A survey on datasets and metrics of textual question answering. arXiv preprint arXiv:2109.12264, 2021.* From the reference, the authors think that the EM is a suitable measure for the DQA task in our experiment.	Add citation [25] in section 3.3.2 and reference reference on the last page.
- In 4.1 If possible, The authors should present the comparison of the quantized models' performance with non-quantized models over an extended period or across a variety of tasks to demonstrate the quantized models' robustness and reliability.	Due to our limited computing resources and limited time to correct the paper, we cannot provide comparisons over an extended period or across a variety of tasks. We make a note of this in our future work.	Last paragraph in section 5 : conclusion – page 8
- In 4.2, The authors should provide additional information on the reasons why quantized models and those utilizing RAG differ in accuracy. This may include an analysis of scenarios or types of questions where each model performs well or poorly.	Upon the investigation of answers mismatched by the pretrained model versus answer matched correctly by pretrained with RAG model, answers containing proper names from specific domains are often found. Hence, the study suggests that when lacking a specific domain knowledge in the context, the pretrained model often generate "hallucinated" answers which is not the correct answer. Therefore, the RAG technique not only helps to eliminate the finetune step but also helps to improve the DQA accuracies. The authors have updated this information in section 4.2.	Section 4.2 – page 8-9

Reviewer 4		
Reviewer's comments	**Point-by-point response**	**Page number**
1. is figure 1 originate from https://int8.io/local-large-language-models-beginners-guide/?, so need to replicate. (also figure 2)	We have redrawn figure 1 as suggested.	Figure 1 and 2 on page 3.
2. In section 2.2, all topics are explained shortly so no need to	We have rewritten section 2.2 into one short paragraph.	Section 2 - Page 4

Reviewer 4		
Reviewer's comments	**Point-by-point response**	**Page number**
separate in sub-section, just separate into paragraph		
3. Section 4.2 the explanation of the results (from table1) make confusing such as, the author shows that "Meanwhile, when applying the RAG technique to each size of pre-trained model, the question answering performance increased significantly, from 7-11% in all model sizes.". however the data shown table 1 decrease from 70 to 68.2	The comparison is between the pretrained model only (the second column) and the pretrained model + RAG model (the third column). It increase from 59% to 70% in the first row (FP16) and increase from 61.2% to 68.2% in the second row (INT8), therefore, RAG enhance the accuracies by 7-11%. The author has modified content in section 4.2 to clarify this.	Section 4.2 Page 7-8.
4. The authors mentioned that they conducted experiments on a computer using only CPU in order to compare model performance and memory usage on various sizes of Quantization. However, they shows that this study conducted a comparison experiment on Kaggle using 2xGPU T4 with 15 GB of memory per GPU.	The phrase "only CPU" in the original paper is misleading, the authors have removed it. All experiments are conducted on Kaggle with GPUs. However, the deployed model is run on HuggingFace's 8vCPU 32GB RAM. We have updated the manuscript to clarify this point.	Introduction section – page 2 Section 5 – conclusion – page 8

Note: The manuscript is shortened to 9 pages based on the ICIC journal 8-9 page limited. (Some redundance contents are removed. Some unnecessary citations and references are removed.)