**Khmer Question-Answering Model by Fine-tuning Pre-trained Model**

Kimleang Ly**1\*** Dona Valy**2** Phutphalla Kong**2**

**1***Graduate student, Master of Engineering in Computer Science, Graduate school, Institute of Technology of Cambodia*

**2***Research and Innovation Center, Institute of Technology of Cambodia, Russian Federation Blvd., P.O. Box 86, Phnom Penh,*

*Cambodia*

Received: 31 July 2024; Revised: 23 August 2024; Accepted: 04 September 2024; Available online: 30 April 2025

|  |
| --- |
| **Abstract:** *As artificial intelligence has grown, a large language model is a model train with a vast quantity of textual data. This model can be tailored to a particular task, such as chatbots, text production, and question-answering. However, most of the existing pre-trained models nowadays were trained with English datasets, leading to limited support and low performance in other languages, especially low-resource languages like Khmer. To address the imbalance, we aim to make a Khmer language model by fine-tuning pre-trained state-of-the-art (SOTA) models, with a focus on question-answering tasks. We propose to use supervised fine-tuning process by providing a labeled dataset to train and utilize the quantization technique along with low-range adaptation for memory-efficient optimization. Moreover, we inject flash attention to make the training process faster. Before we start the experiment, we observed that some SOTA models were not able to recognize Khmer language. To deal with this problem, we do vocabulary expansion. To achieve our experiment, we collect datasets from online sources containing question-and-answer pairs in general knowledge domain. The three decoding strategies including greedy search, beam search, and contrastive search use to select the output tokens to generate text. We use bilingual evaluation understudy (BLEU) as an evaluation metric because it measures the similarity between generated responses and referenced sentences. Through the experiment, we obtained the BLEU score of Gemma 7B fine-tuned model increase from 0.0539 to 0.2863 on greedy search, from 0.0227 to 0.2765 on beam search, and from 0.0009 to 0.2201 on contractive search. The increasing showed that the fine-tuning process enhance the performance of model. This score also indicated that the model can generate the clear response but have grammatical error. The findings of this study contribute to the growing research on applying Khmer language with deep learning techniques to make question-answering. In conclusion, this finding will offer a multitude of benefits across various domains. Their ability to understand natural language makes them invaluable tools for businesses, educators, and researchers.* |
| **Keywords:**  Artificial Intelligence (AI), Large Language Model (LLM), Fine-tuning, Pre-trained Model, Supervised Fine-tuning |

1. INTRODUCTION[[1]](#footnote-1)

Large language models (LLMs) have transformed natural language processing (NLP) by demonstrating remarkable ability in tasks related to text generation, comprehension, and decision-making. Under-resourced languages like Khmer receive less attention from many of the models now in use, which are primarily concentrated on large resource languages like English. Because of its intricate script, extensive morphology, and relatively scarce availability of labeled data, Khmer poses particular difficulties for NLP.

The recent LLMs are the successful product of research and development from language models, starting from the statistical language model, neural language model, pre-trained language model, and LLMs according to [1]. Statistical language model (SLM) also known as n-gram model predict the next token based on the probability of word $w$ by giving previous context. Due to data sparsity, this model cannot capture the diversity and variability of language. The neural language model (NLM) uses the embedding vector to deal with data sparsity and predict output using a neural network. A pre-train language model (PLM) is usually train using a recurrent neural network (RNN) or transformer model with large-scale web text data (unsupervised data) for many tasks and then fine-tune to specific tasks by providing a small amount of label data. LLM is a transformer-based model with a hundred to billion parameters that are pre-train with massive text data. LLM can learn new tasks from a small set of samples present as prompt in inference time, follow the instructions of a new task, and solve a complex task by breaking that task into small steps. There are recent studies related to the multilingual language model and their approach to the fine-tuning process that use for applying in these experiments.

Llama 2-chat were developed and released by [2]. It was a fine-tuned version of Llama 2 that was optimized for conversation use cases. The process of fine-tuning started with pre-training Llama 2 using a public dataset. Next, they implemented supervised fine-tuning, and then the model was iteratively refined using Reinforcement Learning with Human Feedback (RLHF) methodologies. They evaluated the performance of this model on various benchmarks, including MMLU, TriviaQA, Natural Questions, GSM8K, HumanEval and BIG-Benh Hard and achieved score 68.9, 85.0, 33.0, 56.8, 29.9 and 51.2 respectively.

Southeast Asian large language model (SeaLLMs) provided by [3], an innovative series of language models specifically focusing on Southeast Asian (SEA) languages. The dataset utilized in this study includes web-based corpora, journalistic content, and text corpora with expertly curated knowledge. SeaLLMs-v1 was built upon the Llama-2 model followed by four stages: in the first process, they conducted continuous pre-training from the base Llama-2 model with an extended vocabulary tailored for SEA languages. After that, they fine-tuned the model in a novel hybrid paradigm with a mixture of multilingual pre-training data and English-dominant instruction fine-tuning data. The following stage subsequently fine-tuned the model on a balanced and custom-built multilingual supervised fine-tuning (SFT) dataset. Finally, they conducted self-preferencing alignment optimization using the SeaLLM model itself, without relying on human annotators or more powerful LLMs. They evaluated the performance of model based on: the category of question-answer tasks and language. This model outperformed ChatGPT-3.5 in non-Latin languages, such as Thai, Khmer, Lao, and Burmese, by large margins while remaining lightweight and cost-effective to operate.

Phi, small language models (SLMs), introduced in [4] that were pre-trained with the CodeTextBook dataset to achieve the base model and then used the CodeExercises dataset for fine-tuning. To create a high-quality dataset, they cleaned their dataset by filtering it using a transformer-based classifier. To ensure the generated code dataset was diverse and not redundant, they included randomness in the words chosen from a fixed vocabulary in the prompts to achieve a diverse dataset. They trained their model with a 2048 input length and used the CodeGen tokenizer, the AdamW optimizer, a linear learning rate scheduler, attention dropout of 0.1, and also utilized the DeepSpeed technique. They evaluated the output of their model with LLM grading, using GPT-4 as a grader, which involved comparing the results from the model with the ground truth and then assigning a score based on how well it matched the real results. Lastly, they noted that high-quality data significantly improved the efficiency of model performance because it provided clear, concise, and balanced examples. They also addressed the limitations of this model compared to other models, stating that it focused only on Python code and lacked knowledge of domain-specific content and diversity in terms of style and programming languages.

The impact of vocabulary size on model and output generation were researched in [5]. The author conducted three experiments. First, they experimented on the number of tokens (5B, 25B, 50B, 100B, 250B, 500B) that used for training tokenizer for three models such as GPT-4, Punct and Llama 1.5B and they found a large difference between Punct and Llama when trained with 5B tokens but this difference almost fades and invert after 50B tokens. Second experiment, they studied on the size of tokenizer. They tested four sizes of tokenizer (32k, 64k, 128k, 256k) on GPT-4 model then evaluated by calculating the Pearson correlation coefficient between tokenizer vocabulary size and HumanEval. The result in correlation coefficient of -0.13 with p-value of 0.87. According to this result, the author concluded that the larger vocabulary sizes were ineffective with model performance. The last experiment studied on the method of updating tokenizer, they compared two methods: Fast Vocabulary Transfer (FVT) and extending the existing tokenizer. They applied these two techniques on GPT-4 model and the result, FVT improved on downstream task while tokenizer extension had less impact on the performance of the model. In conclusion, the author addressed that a larger vocabulary size helped models understand a wider range of text but increased computational cost and time, with only a minimal impact on model performance.

This study focuses on investigating the effectiveness of fine-tuning pre-trained models including, Gemma 7B, Llama-3 8B, Mistral 7B v3, Phi 2, Qwen-2 7B, and SeaLLM v2.5 by applying a small dataset size to adapt it to question-answering tasks and evaluate the result using BLEU. This experiment also applies two techniques like QLoRA and Flash attention to make the smooth training process without facing out-of-memory problem.

In the subsequent sections of this paper, we will provide an overview of the methodology in section 2, section 3 describes the approaches used in these experiments, the result and discussion are in section 4, and end with section 5 is about the conclusion.

# METHODOLOGY

## Data Collection

The text dataset utilized in this research collect from an online website specialized in the general knowledge domain, comprising 10,698 pairs of questions and corresponding answers. This dataset contains various topics and linguistic variations in the Khmer language. We separate the dataset into three subsets: training set, testing set, and validation set. The training set comprises most of the dataset, consisting of 7,488 question-answer pairs to serve as primary data for model training. The testing set consists of 1,605 pairs used for evaluating the generalization of the model and its performance. The validation set contains 1,605 data pairs used to evaluate the performance of the model during the training process.

## Data Preparation

During the training phase, the model performance and efficacy largely depend on the quality of the data. In this study, the quality and variety of the dataset use to train the models improve by applying two crucial data-cleaning techniques: data filtering and deduplication. The data preparation cycle starts with data filtering to get rid of noise and unimportant information that could affect model training. To remove that noise, first, identify the regular expression pattern like this example, “[៙៚៖!?។៕\u17d8\u200b~]” then iterate over the dataset to remove it from each data point. To stop the model from learning duplicate patterns, basic deduplication implements by eliminating redundant questions along with their answers.

## Model Architectures

Most large language models (LLMs) are built based on decoder-only transformer architecture that contains a hundred to billion parameters that can understand, generate, and solve a complex task as well. [Fig. 1](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.susovttaldo) shows the general architecture of the decoder-only model [6].This auto-regressive model generally has four main components as below:

* **Positional Encoding**. Transformer is known as no recurrent model however the order of each input plays a crucial part, they add information about the position either relative or fixed to each token in input sequence, and have the same dimension as embedding.
* **Attention Layer**. Allow model to learn the relation between each word in the input sequence. There are three main variables: key, query, and value.
* **Layer Normalization**. Reduce the time of computation by standardizing the activity of the neuron according to [7].
* **Feed-Forward Network (FFN)**. This network consists of a ReLU activation function that is in between two linear transformations. It applied to each input position independently.

In this study, we experiment with six state-of-the-art auto-regressive models that altered the main layers to achieve optimal performance.

Gemma, a model developed by [8], suitable for many tasks like text generation, question-answering, text summarization, etc. This model trained with a massive English dataset that mainly collected from web documents, code, mathematics, etc. This model consisted of two variants 2B and 7B. The values of parameters used in these models are shown in Table 1. The different key component of these two models is the attention layer.

* **Multi-Query Attention (MQA)**. Share the same key and value across the different queries for each attention layer introduced in [9]. This technique aims to reduce the memory bandwidth. Gemma 2B was implemented using this method as seen in Fig. 2.
* **Multi-Head Attention (MHA).** Each attention layer that runs in parallel uses a different key, query, and value [6]. This approach was employed with Gemma 7B model as seen in Fig. 2.

The same technique is utilized in these two variants:

* **Rotary Positional Embeddings (RoPE)**. Instead of using the original position embedding (absolute positional embedding), they implemented RoPE to enhance the performance of the architecture. [10] converts positional data so the model can process both the relative and absolute distances between tokens. This is accomplished by using a rotating mechanism, in which a rotation in the embedding space corresponds to each location in the sequence.
* **GeGLU Activations**. Is a nonlinear function, [11] is used in feed-forward networks instead of ReLU activation function.
* **Root Mean Square Layer Normalization (RMSNorm)** Rather than normalizing the output, [12] normalize each transformer sub-layer input to increase training stability.

Llama3 model was introduced in [[13]](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.b5ezl2i149ec), customized components of the decoder-only transformer model to maximize performance. This included using RMSNorm, a simplified version of LayerNorm, replacing absolute position embeddings by RoPE to enhance input position information, injecting grouped query attention mechanism into attention layer, and using SwiGLU activation in FFN. The information on parameters in this model is shown in Table 1.

* **Grouped Query Attention (GQA)**. Divided a query head into *G* groups and each group shared the same head and key according to [[14]](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.bjv91fiyi273). GQA is an intermediate approach between Multi-Head Attention (MHA) and Multi-Query Attention (MQA), achieving a balance between quality and speed while generating tokens as seen in Fig. 2.
* **SwiGLU Activation**. To improve the performance of model, they replaced ReLU with the SwiGLU activation function.

The Mistral 7B model [[15]](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.6m6qcvn6l7ms) was a decoder-only based on the transformer architecture which consists of three main components: Root Mean Square Normalization (RMS Norm), Self-Attention layer, and Feed-forward networks that used a Sigmoid Linear Unit (SiLU) as an activation function. In addition, the details of model configuration of decoder stack and attention mechanism parameters show in Table 1. To make a large language model more efficient and high-performance, they introduced a few novel techniques below

* **Group Query Attention (GQA)**. This method is already described in Llama model.
* **Sliding Window Attention (SWA)**. In each attention layer, the information can move forward by W tokens and each token can attend at most W tokens from the previous layer. The token outside the window was still indirectly related to the next token prediction.
* **Rolling Buffer Cache**. Data for position *i* is stored in cache position *i % W*. The cache has a fixed capacity of *W*. The capacity of the cache stops growing and previous values are overwritten when position *i* exceeds *W*.
* **Pre-fill and Chunking**. Process by chunking prompt into smaller pieces if long and prefilling *(k, v)* cache with each chunk then compute attention over chunk and cache. The chunk size can be equal to the size of the sliding window.

Phi, a series of small language models that consisted of four versions: Phi 1, Phi 1.5, Phi 2, and Phi 3. In the experiment, we chose the Phi 2 model architecture which contains 32-layer stacks, and for configuration of this model show in Table 1. Phi 2 adjusted the based architecture with other approaches like using RoPE as a replacement of absolute position encoding and also changing from ReLU activation function to GeLU.

Alibaba Cloud developed a series of language models called Qwen [16]. Qwen models relied on auto-regressive model architecture and also trained in many languages including Khmer. To achieve a high performance of generating output, they improved FFN by using SwiGLU activation which is a combination of Swish and Gated Linear Unit, adding the bias to QKV attention layers to enhance the ability of model prediction, group query attention (GQA), RMSNorm, and RoPE. Additionally, they improved the tokenizer to adapt to multiple languages. More information on this model shows in Table 1.

SeaLLM-v2.5 7B is a model that continues pre-trained and fine-tuned from a backbone Gemma 7B model without causing a decline in high-resource language performances [3]. Prioritizing local cultural and legal conventions, customs, stylistic preferences, and cost-effectiveness is a key focus in the construction of SeaLLMs. This architecture and configuration of model were the same as Gemma 7B.

# EXPERIMENT

## Expand Vocabulary

Some of the LLMs vocabulary does not have or less contain Khmer tokens that might cause misunderstandings among models. To end this problem, we add the new Khmer words/sub-words that exist in the No Language Left Behind (NLLB) to the original LLM word list. NLLB is a translation model that uses the Byte-Pair Encoding approach to tokenize input data and it was trained with over 200 languages including Khmer.  [Fig. 3](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.k16omkbrar7g) shows the before and after adding Khmer token to Llama 3 8B model. After the new token is added, we resize model embedding and output dimension to the current vocabulary size and then retrain the embedding layer during the fine-tuning process.

## Efficient Fine-tuning Technique

To make a large language model more efficient and performing, fine-tuning is the best way to do that.  However, fine-tuning a large model is expensive. For example, if we fully fine-tune on Llama3-8B model that one parameter is stored in 16 bits then you need more than 100 GB of GPU memory. To overcome this issue, [17] introduces a technique called QLoRA that uses a novel high-precision technique to quantize a pre-trained model to 4-bit, then adds a small set of learnable Low-rank Adapter weights. This technique is a combination of quantization and low-range adaptation.

Quantization is a process of converting the model weight that is represented as a data type with more bits to fewer bits [17]. For example, compressing a 16-bit floating point (FP16) tensor to Int4 tensor with range [-127, 127] followed by Eq.1 and Eq.2 shows how to reverse it to the original data type.

$x^{int4}=round\left(\frac{127}{absmax(x^{FP16})}X^{FP16}\right)=round(c^{FP16}.X^{FP16})$ (Eq. 1)

$dequant\left(c^{FP16}, X^{Int4}\right)= \frac{X^{Int4}}{C^{FP16}}=X^{FP16}$ (Eq. 2)

Where:

c = quantization constant

*X* = input tensor

In addition, they also provide multiple techniques like 4-bit normal float (NF4), double-quantization, and paged optimizers to avoid low performance and cut-off memory allocation.

* **4-bit NormalFloat (NF4)**. Normalize model weight that is stored in a 4-bit float to value between [-1, 1].
* **Double Quantization (DQ)**. The procedure for further memory savings by quantizing the quantization constants *c* in Eq.1 above.
* **Paged Optimizers**. This function is used to allocate paged memory for the optimizer states, which are then automatically moved from GPU memory when the GPU runs out of memory to CPU RAM and returned to GPU memory when the optimizer update step requires it.

To increase the efficiency and minimize the number of trainable parameters of fine-tuning. The low-range adaptation (LoRA) approach [18] represents the trainable weight with two smaller matrices via low-range decomposition. Training these new matrices with new data while the remaining weights are fixed. Both original and adapted weights are combined to get the final results. [Table 2](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.iir67nhqa7i4) describes QLoRa setting parameters for each model experiment. The variable *r* is the rank of update matrices and *lora\_alpha* is the scaling factor of LoRA.

**Table 1**. The value of each original model parameter

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parameters | Gemma 7B | Llama3 8B | Mistral 7B v0.3 | Phi 2 | Qwen v2 7B |
| d\_model | 3,072 | 4,096 | 4,096 | 2,048 | 3,584 |
| num\_layers | 28 | 32 | 32 | 32 | 32 |
| hidden dims | 49,152 | - | 14,336 | 2,560 | 4,096 |
| num\_heads | 16 | 32 | 32 | 32 | 32 |
| num\_kv\_heads | 16 | 8 | 8 | 32 | 32 |
| head size | 256 | 256 | 128 | 32 | 256 |
| vocab\_size | 256,000 | 128,256 | 32,768 | 51,200 | 151,936 |

**Table 2**. QLoRA configuration for each model experiment

|  |  |  |
| --- | --- | --- |
| Model | Quantization | LoRA |
| use 4-bitnormal float | double-quantization | r | lora\_alpha |
| Gemma 7B | ✓ | ✓ | 128 | 256 |
| Llama 3 7B | ✓ | ✓ | 128 | 256 |
| Mistral v0.3 | ✓ | ✓ | 128 | 512 |
| Seallm v2.5 | ✓ | ✓ | 8 | 16 |
| Phi 2 | ✓ | ✓ | 128 | 256 |
| Qwen 2 7B | ✓ | ✓ | 128 | 512 |

Moreover, to make the training and inference faster. We add flash attention to the fine-tuning process. Flash attention was introduced in [19] for solving a problem of standard attention that stores, reads, and writes keys (K), queries (Q), and values (V) using high bandwidth memory (HBM). In contrast, flash attention load Q, K, and V once then combine the attention mechanism operator and write them back. [Fig. 4](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.s6pwdwensa55) shows the comparison of standard attention and flash attention processes. It also shows parameters loading from the memory to computation.

## Supervised Fine-tuning Process

Fine-tuning is a process of training additional data in a specific domain with a pre-trained model to get a new model that has knowledge related to that domain. Supervised fine-tuning (SFT) refers to a process of fine-tuning models within label data. We apply the same flow to experiment with each model but have different settings on some variables shown in [Table 3](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.vk7su47ohopn). The SFT process starts with loading the training dataset which has two columns named “prompt” and “completion”. We concatenate each prompt and completion in the training set and separate them by the special token of each model. For example, the prompt format of Mistral 7b model <s>[INST]តើប្រទេសកម្ពុជាមានទំហំប៉ុន្មាន[/INST]181,035គីឡូម៉ែត្រការ៉េ</s>. Then we load the tokenizer. If it can tokenize Khmer input text, we do not add any new token otherwise we do vocabulary expansion. Padding is enabled in the tokenized sequences, with the padding token set to *eos\_token* (end-of-sequence token) placed on the right side. This ensures uniform input sequence lengths required for efficient batch processing during model training. Truncation also enables handling sequences longer than the determined maximum length. Next, we load the model by applying a quantization technique to downcast a pre-trained model weight to 4-bit precision and flash attention. After that, we set training arguments such as the max input sequence length equal to 1,000, batch size 100, training batch size per device to 1, and learning rate to 2e-4. Additionally, we employ LoRA to all attention layers and add two layers such as the embedding layer and the output layer, in case the model needs to process vocabulary expansion. During the training, we evaluate the performance of the model on the validation dataset every 500 training steps. Moreover, we inject an early stopping callback to stop training if no improvement. Lastly, we merge the original model with a trained adapter to get a fine-tuned model.



**Fig. 1.** The decoder-only transformer-based model architecture with N=6 layers stack by default



**Fig. 2**. The diagram of three attention-layer approaches

**Table 3**. The difference in training parameters configuration of each original model

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Vocabulary expansion | Number of epochs | Optimizer type |
| Gemma 7B | 🗶 | 10 | paged adamw 32bit |
| Llama 3 8B | ✓ | 10 | paged adamw 8bit |
| Mistral v0.3 | 🗶 | 10 | adamw bnb 8bit |
| Seallm v2.5 | 🗶 | 4 | paged adamw 8bit |
| Phi 2 | ✓ | 10 | paged adamw 8bit |
| Qwen 2 7B | 🗶 | 10 | adamw bnb 8bit |



**Fig. 3**. An example of Llama 3 8B before and after adding Khmer tokens to their vocabulary. The original broke down the input data into many unknown tokens and only two tokens were recognized as “ា” and “្” but after merging Khmer tokens into its vocabulary, this model can now tokenize Khmer language into words and sub-words.



**Fig. 4**. The difference between standard attention and flash attention access and load parameters from the memory to conduct computation.

## Decoding Strategies

Decoding is a process of selecting output tokens to generate text. We use deterministic methods (greedy search and beam search) and contrastive search. [Table 4](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.yd7eyor1vs8s) shows the configuration of decoding strategies used in the experiment.

Greedy search chooses the high probability token at each time step as the next token in the output sequence without considering the whole effect on output. This technique is fast in generating text but it can lose the better token which is hidden behind the lower probability of the next token. Eq. 3 is a formula for finding the next token at time step *t*. Fig. 5 is an example sketch of a greedy search that chooses the next token.

$w\_{t}=argmax\_{w}P\left(w|w\_{1:t|t-1}\right)$ (Eq. 3)

Where:

w = token

t = current time step

Beam search is a process of finding the sequence that has the highest probability in total. This process is repeated until an end-of-sequence token appears or a predetermined maximum sequence length is reached. [Fig. 6](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.xr3e3t7urxc5) is a beam search diagram with a number of beams equal to 2. The number of beam increase led to more coherent and grammatically correct outputs. However, it requires high computational cost and generate repetitive output token.

Contrastive search generates output based on the set of most probable tokens that model predicted and sufficiently discriminative with the preceding context at each decoding step [20]. The process of contrastive search begins with generating a set of the next tokens at each time step is generally done by choosing the *top-k* tokens from output of model. After that, calculate a degeneration penalty of each candidate token by measuring the similarity of representation of candidate token and the previous representation of generated token. This computation aims to avoid the lack of diversity or repetition of generated tokens. Next, update the probability of each candidate token then select the next token to the generated text based on the adjustment from the previous step. Next step, update the current context by adding the selected token, and it will be used to generate the next token, this process ends when it meets the stopping criteria or reaches the maximum output length.



**Fig. 5.** Example of how greedy search chose the next token based on the probability of it. The word “woman” has a higher probability of 0.5 than other words like “car” and “boy”. So, the word “woman” has been added to the output sequence and this process will repeat until it meets the stop criteria



**Fig. 6**. The sample sketch of how beams search that number of beams equal to 2. Besides the most likely hypothesis ("This", "woman") ("This", "boy"), beam search also keeps track of the second most likely one ("This", "boy") at time step 1 and in time step 2, beam search finds that the word sequence ("This", "boy", "sad") has 0.27 a higher probability than ("This", "woman", "laugh") which has 0.25.

**Table 4**. Decoding techniques configuration parameter

|  |  |  |  |
| --- | --- | --- | --- |
| Parameters | Greedy search | Beam search | Contractive search |
| num\_beams | 1 | 7 | - |
| top\_k | - | - | 30 |
| penalty\_alpha | - | - | 0.6 |
| do\_sample | False | - | - |
| early\_stopping | - | True | - |

## Evaluation Metric

The similarity of n-grams, or sequences of *n* words, between the reference text and the predicted text is measured using the Bilingual Evaluation Understudy (BLEU) score to assess the correctness of the model outputs. A number nearer 1 denotes a better degree of alignment and resemblance between the anticipated and reference texts. The BLEU score runs from 0 to 1. Before discussing the calculation of the BLEU score, let's review n-gram and precision. A group of n-words or tokens used in a phrase is called an n-gram. An n-gram of two, for example, denotes a sequence of two words, such as "hello world" or "ask me," where each word is counted according to the space separator. The number of n-gram words that overlap between the target sentence and the predicted sentence is known as precision. The formulas for the Brevity Penalty (BP) and the Geometric Average of the Precision scores (GAP) [21], which are required to calculate the BLEU score, are provided in Eq. 4 and Eq. 5, respectively. When the length of the candidate translation equals the length of any reference translation, BP will be 1.0. Eq. 6 displays the BLEU scoring formula.

In the experiment with Khmer language, no space between each word in a sentence. Splitting the input sentence before calculating the score based on n-gram. The n-gram in this use case is a number of sub-words and words because the tokenizer tokenizes input sequence into sub-words and words as an example shown in Fig. 7.

$BP=\left\{\begin{array}{c}1 if c>r\\e^{(1-{r}/{c) }} if c\leq r\end{array}\right.$ (Eq. 4)

Where:

c = count of words in candidate/prediction sentence

r = count of words in reference sentence

$GAP\left(N\right)=\left(\sum\_{n=1}^{N}w\_{n}logp\_{n}\right)$ (Eq. 5)

Where:

N = number of n-grams

$w\_{n}$ = positive weight

$p\_{n}$ = n-gram precision

$BLEU score=BP.GAP(N)$ (Eq. 6)

Where:

BP = brevity penalty calculated in Eq. 4

GAP = geometric average of precision score calculates in Eq. 6



**Fig. 7**. An example of Byte pair encoding tokenizes Khmer language into words and sub-words

# RESULTS AND DISCUSSION

The comparison result of BLEU score between model baseline and fine-tuned model is shown in [Table 5](https://docs.google.com/document/d/17_FkHflLtm6LB7-eeSZwWDPaun46UgBJ/edit#heading=h.to1u62k41xxn). According to the result, each fine-tuned model demonstrates improvement in Khmer text generation compared to the base model. For instance, the BLEU score of Gemma 7B model increased from 0.0539 to 0.2863 on greedy search, from 0.0227 to 0.2765 on beam search, and from 0.0009 to 0.2201 on contrastive search. The lower BLEU score of the based model reveals that the initial model trained with a small amount of Khmer data. The increasing in BLEU score demonstrated that the fine-tuning process enhanced the ability of model to produce output. However, the obtained BLEU score of 0.2863 indicated that the model was able to generate a clear response that had similarity to the reference text but contained grammatical errors according to the standard score interpretation. In addition, the higher score on greedy search than beam search and contrastive search showed that the greedy search, which selects the most probable token at each step that has a better n-gram overlap with reference. In contrast, beam search and contrastive search explore more diverse output sequences that may introduce some noise or less fluent text that can negatively impact the BLEU metric. The example of each fine-tuned model output is shown in Table 6.

# CONCLUSIONS

In this final section, the dataset with a diversity of question-and-answer pairs in the general knowledge domain was successfully prepared as a baseline for conducting experiments in six different decoder-only transformer-based models. The Gemma 7B model obtained the higher BLEU score among another model, 0.2863 on greedy search, 0.2765 on beam search and 0.2201 on contrastive search. The experiment demonstrated that the small dataset of training model by fine-tuning models improved the ability of understanding and generated Khmer text based on the BLEU score on three difference decoding strategies. However, the BLEU score indicated the moderate performance of model generated output. Moving forward, to improve response of fine-tuned models in this study by implementing the direct preference optimization approach (DPO) that need to collect human preference dataset. The evaluation with another metrics should be conduct to get more insight on model output, including Human evaluation, F1-score and Perplexity. Last but not least, to contribute to Khmer language model research and development, free access to the dataset and model for the next researcher who interested in this field.

**Table 5**. Fine-tuned (left) and original (right) models result in BLEU

|  |  |  |  |
| --- | --- | --- | --- |
| Models | Greedy search | Beam search | Contrastive search |
| Gemma 7B | **0.2863** | 0.0539 | **0.2765** | 0.0227 | **0.2201** | 0.0009 |
| Llama 3 8B | 0.1947 | 0.0417 | 0.1774 | 0.0342 | 0.1852 | 0.0367 |
| Mistral v3 | 0.2129 | 0.0779 | 0.2166 | 0.0794 | 0.1374 | 0.0299 |
| Seallm v2.5 | 0.1977 | 0.1975 | 0.1938 | 0.1596 | 0.0871 | 0.0015 |
| Phi 2 | 0.0129 | 0.0047 | 0.0147 | 0.0017 | 0.0124 | 0.0031 |
| Qwen v2 7B | 0.2184 | 0.1310 | 0.2204 | 0.0719 | 0.1807 | 0.0166 |

**Table 6**. Example of question and answer from each fine-tuned model

|  |  |
| --- | --- |
| Question | តើប្រាសាទសំបូរព្រៃគុកនេះក្នុងបញ្ជីបេតិភណ្ឌពិភពលោកនៅថ្ងៃខែឆ្នាំណា? |
| Answer | ប្រាសាទសំបូរនៃព្រៃគុកចុះក្នុងបញ្ជីបេតិកភណ្ឌពិភពលោកនៅថ្ងៃទី០៨ ខែកក្កដា ឆ្នាំ២០១៧ |
| Gemma 7B | ប្រាសាទសំបូរព្រៃគុកនេះក្នុងបញ្ជីបេតិកភណ្ឌពិភពលោកនៅថ្ងៃទី១២ខែកក្កដាឆ្នាំ២០១៧រៀបចំប្រគល់បេតិកភណ្ឌពិភពលោកនៅក្រុងក្រាកូអានប្រទេសប៉ូឡូញ |
| Llama3 8B | ប្រាសាទសំបូរព្រៃគុកនេះក្នុងបញ្ជីបេតិកភណ្ឌពិភពលោកនៅថ្ងៃ ០៣ កក្កដា ២០១៧ |
| Mistral 7B v0.3 | ប្រាសាទសំបូរព្រៃគុកនេះក្នុងបញ្ជីបេតិភណ្ឌពិភពលោកនៅថ្ងៃទី៨ ខែកក្កដា ឆ្នាំ ២០១៧ ខេត្តកំពង់ធំ ប្រទេសកម្ពុជា |
| Seallm 7B v2.5 | ប្រាសាទសំបូរព្រៃគុកនេះក្នុងបញ្ជីបេតិភណ្ឌពិភពលោកនៅថ្ងៃទី ១៩ ខែកក្កដា ឆ្នាំ ២០០៨ ដោយសារមានការរីកចម្រើននៃការទេសចរណ៍ |
| Phi 2 | ▁ចាំ |
| Qwen v2 7B | ប្រាសាទសំបូរព្រៃគុកនេះក្នុងបញ្ជីបេតិកភណ្ឌពិភពលោកនៅថ្ងៃទី០៨ខែកក្កដាឆ្នាំ២០១៧នៅទីក្រុងកេបិចប្រទេសកាណាដា ប្រាសាទនេះមានទីតាំងនៅក្នុងភូមិសំបូរ ឃុំសំបូរ ស្រុកប្រាសាទសំបូរ ខេត្តកំពង់ធំ ដែលមានចម្ងាយ៣២គម ពីទីរួមខេត្តកំពង់ធំ និងចម្ងាយ១២០គម ពីទីក្រុងភ្នំពេញ ប្រាសាទសំបូរព្រៃគុក ត្រូវបានកំណត់អោយជាបេតិកភណ្ឌពិភពលោកនៅថ្ងៃទី០៨ ខែកក្កដា ឆ្នាំ២០១៧ ដែលមានទំហំ ២៦២ ហិចតា និងកម្ពស់១៣២ម៉ែត្រ |
| Question | តើប្រជាពលរដ្ឋត្រូវមានករណីកិច្ចអ្វីខ្លះចំពោះសង្គមជាតិ? |
| Answer | ប្រជាពលរដ្ឋម្នាក់ៗត្រូវមានករណីយកិច្ចចាំបាច់ក្នុងការចូលរួមយ៉ាងសកម្ម ក្នុងការកសាងប្រទេសជាតិទាំងវិស័យសេដ្ឋកិច្ច សង្គមកិច្ច វប្បធម៌ និងនយោបាយរបស់ប្រទេសឱ្យបានខ្លាំងក្លាដូចជាត្រូវមានតួនាទីបង់ពន្ធអាករជូនរដ្ឋមានតួនាទីត្រូវគោរពច្បាប់រដ្ឋមានតួនាទីត្រូវសិក្សាវិជ្ជាផ្សេងៗត្រូវបំពេញតួនាទីការពារជាតិបង្កបង្កើនផលរក្សាប្រពៃណីទំនៀមទំលាប់ល្អរបស់ជាតិរក្សាការពារបរិស្ថាន និងសម្បត្តិវប្បធម៌ជាតិត្រូវប្រើសិទ្ធិចូលរួមបោះឆ្នោតជ្រើសរើស អ្នកតំណាងរាស្រ្ត និងអ្នកដឹករបស់ខ្លួនក្នុងការគ្រប់គ្រង និងអភិវឌ្ឍប្រទេស |
| Gemma 7B | ប្រជាពលរដ្ឋត្រូវមានករណីកិច្ចចំពោះសង្គមជាតិដូចតទៅគោរពនិងការពាររដ្ឋធម្មនុញ្ញនិងគោរពប្រតិបត្តិនានាដែលបានកំណត់ដោយរដ្ឋធម្មនុញ្ញគោរពនិងការពាររបបរាជានិយមនិងរបបសាធារណរដ្ឋនិយមគោរពនិងការពារទំនៀមទំលាប់ប្រពៃណីនិងវប្បធម៍ជាតិគោរពនិងការពារបូររាជទ្រព្យនិងមរតកវប្បធម៍ជាតិគោរពនិងការពារប្រយោជន៍រួមនិងការទទួលខុសត្រូវចំពោះសង្គមជាតិគោរពនិងការពារនូវការអនុវត្តច្បាប់នានាគោរពនិងការពារសមត្ថកិច្ចនិងការទទួលខុសត្រូវរបស់រដ្ឋអំណាចគោរពនិងការពារនូវការគោរពសិទ្ធិមនុស្សនិងកុមារ |
| Llama3 8B | ប្រជាពលរដ្ឋត្រូវមានករណីយកិច្ចចំពោះសង្គមជាតិដូចតទៅគោរពសិទ្ធិសេរីភាពរបស់ប្រជាពលរដ្ឋអោយរួចផុតពីការធ្វើទុក្ខទោសពីអ្នកដទៃ |
| Mistral 7B v0.3 | ប្រជាពលរដ្ឋត្រូវមានករណីយកិច្ចចំពោះសង្គមជាតិដូចជាប្រជាពលរដ្ឋត្រូវគោរពច្បាប់នានារបស់ប្រទេសជាតិប្រជាពលរដ្ឋត្រូវចូលរួមចំណែកក្នុងកិច្ចការរដ្ឋប្រជាពលរដ្ឋត្រូវគោរពសិទ្ធិមនុស្សប្រជាពលរដ្ឋត្រូវគោរពសេរីភាពរបស់ប្រជាពលរដ្ឋ |
| Seallm 7B v2.5 | ប្រជាពលរដ្ឋត្រូវមានករណីកិច្ចចំពោះសង្គមជាតិដូចជា ករណីកិច្ចសេរីភាព ករណីកិច្ចសិទ្ធិមនុស្ស ករណីកិច្ចការងារ ករណីកិច្ចអាជីវកម្ម ករណីកិច្ចអាជីវកម្មកសិកម្ម ករណីកិច្ចអាជីវកម្មបច្ចេកវិទ្យា ករណីកិច្ចអាជីវកម្មមហាជីវកម្ម ករណីកិច្ចអាជីវកម្មកសិកម្ម ករណីកិច្ចអាជីវកម្មបច្ចេកវិទ្យា  |
| Phi 2 | រយៈ the khmer question below |
| Qwen v2 7B | ប្រជាពលរដ្ឋត្រូវមានករណីកិច្ចចំពោះសង្គមជាតិដូចជាត្រូវចូលរួមបំបាត់ភាពក្រីក្រ និងការអត់ឃ្លានរបស់ប្រជាពលរដ្ឋរស់នៅក្នុងសង្គម ត្រូវចូលរួមបំបាត់អំពើពុករលួយ និងសេពគ្រឿងញៀន ត្រូវចូលរួមបំបាត់ការជួញដូរសត្វព្រៃ និងឬសីលធម៌ ត្រូវចូលរួមបំបាត់ការប្រើប្រាស់គ្រឿងញៀន និងសារធាតុពុលត្រូវចូលរួមបំបាត់ការរើសអើង និងការបង្កើតសង្គមចំណីអាហារ |

# REFERENCES

* 1. S. Minaee et al., “Large Language Models: a survey,” arXiv.org, Feb. 09, 2024. <https://arxiv.org/abs/2402.06196>
	2. H. Touvron et al., “Llama 2: Open foundation and Fine-Tuned chat models,” arXiv.org, Jul. 18, 2023. https://arxiv.org/abs/2307.09288
	3. X.-P. Nguyen et al., “SEALLMS -- Large Language Models for Southeast Asia,” arXiv.org, Dec. 01, 2023. https://arxiv.org/abs/2312.00738
	4. S. Gunasekar et al., “Textbooks are all you need,” arXiv.org, Jun. 20, 2023. https://arxiv.org/abs/2306.11644
	5. G. Dagan, G. Synnaeve, and B. Rozière, “Getting the most out of your tokenizer for pre-training and domain adaptation,” arXiv.org, Feb. 01, 2024. https://arxiv.org/abs/2402.01035
	6. A. Vaswani et al., “Attention is All you Need,” arXiv (Cornell University), vol. 30, pp. 5998–6008, Jun. 2017, [Online]. Available: https://arxiv.org/pdf/1706.03762v5
	7. J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” arXiv.org, Jul. 21, 2016. https://arxiv.org/abs/1607.06450
	8. G. Team et al., “GemMa: Open models based on Gemini research and technology,” arXiv.org, Mar. 13, 2024. https://arxiv.org/abs/2403.08295
	9. N. Shazeer, “Fast Transformer Decoding: One Write-Head is All You Need,” arXiv.org, Nov. 06, 2019. https://arxiv.org/abs/1911.02150
	10. J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, “RoFormer: Enhanced transformer with Rotary Position Embedding,” Neurocomputing, vol. 568, p. 127063, Feb. 2024, doi: 10.1016/j.neucom.2023.127063.
	11. N. Shazeer, “GLU variants improve transformer,” arXiv.org, Feb. 12, 2020. https://arxiv.org/abs/2002.05202
	12. B. Zhang and R. Sennrich, “Root mean square layer normalization,” arXiv.org, Oct. 16, 2019. https://arxiv.org/abs/1910.07467
	13. H. Touvron et al., “LLAMA: Open and Efficient Foundation Language Models,” arXiv.org, Feb. 27, 2023. https://arxiv.org/abs/2302.13971
	14. J. Ainslie, J. Lee-Thorp, D. J. Michiel, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, “GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints,” arXiv.org, May 22, 2023. https://arxiv.org/abs/2305.13245
	15. A. Q. Jiang et al., “Mistral 7B,” arXiv.org, Oct. 10, 2023. https://arxiv.org/abs/2310.06825
	16. J. Bai et al., “Qwen Technical Report,” arXiv.org, Sep. 28, 2023. https://arxiv.org/abs/2309.16609
	17. T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “QLORA: Efficient Finetuning of Quantized LLMS,” arXiv.org, May 23, 2023. https://arxiv.org/abs/2305.14314
	18. T. Dao, “FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning,” arXiv.org, Jul. 17, 2023. https://arxiv.org/abs/2307.08691
	19. Y. Su and N. Collier, “Contrastive search is what you need for neural text generation,” arXiv.org, Oct. 25, 2022. https://arxiv.org/abs/2210.14140
	20. K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU,” ACL, Jan. 2001, doi: 10.3115/1073083.1073135.

1. \* Corresponding author: Kimleang LY *E-mail: lykimleang456@gmail.com; Tel: +855-89 251 962*  [↑](#footnote-ref-1)