# Fine-Tuning mGPT-1.3B-Armenian For Armenian Question-Answering Task

Astghik Kostanyan, Anahit Zakaryan, Anahit Navoyan

American University of Armenia

DS 235: Generative AI

Dr. Davit Abgaryan

May 17, 2024

# Fine-Tuning mGPT-1.3B-Armenian For Armenian Question-Answering Task

## Introduction

Many models perform well on various tasks in rich-resource languages, however, languages like Armenian, which are low-resource, face numerous challenges in the same tasks. These low-resource languages often lack sufficient data in the training corpus of the model, and fine-tuning for a specific task can significantly improve their performance, especially when the pretrained model has limited data for such languages.

The objective of this project is to fine-tune the LLM model for the Armenian language downstream task. By doing this, we hope to achieve better results and improve the model's performance in generating accurate Armenian text, thereby contributing to the Armenian language community. More specifically, we aim to fine-tune the model for general knowledge-based question-answering tasks.

GPT is one of the available models that supports the Armenian language and generates Armenian text. However, since it is not open-source and fine-tuning GPT is computationally expensive, we have decided to experiment with a model that is similar to GPT but is open-source, has fewer parameters, and can be easily used by anyone. This model is Multilingual GPT fine-tuned on Armenian (ai-forever/mGPT-armenian), available in the Huggingface library.

In the paper “mGPT: Few-shot Learners Go Multilingual,” Shliazhko et al. (2022) introduce a family of autoregressive GPT-like models with 1.3 billion parameters, trained on 61 languages using Wikipedia and the Colossal Clean Crawled Corpus. This multilingual GPT model, mGPT, has the architecture of the GPT-3 model, and the sources and sparse attention mechanism are taken from GPT-2. Armenian is one of the 61 languages included, and they have developed a separate monolingual model, mGPT-1.3B-Armenian, with 1.3 billion parameters derived from the base mGPT-XL (1.3B) model.

## Methodology

### 1. Data Collection

The data used for fine-tuning was gathered through several steps, combining the WebQuestions dataset with GPT-generated question-answer pairs.

The WebQuestions dataset is a question-answering dataset that uses Freebase as the knowledge base and contains 6,642 question-answer pairs. The original split includes 3,778 examples for training and 2,032 for testing. The questions mainly focus on single named entities and cover popular topics asked on the web, ranging from entertainment to science. For this project, only the training data was used. The final dataset consisted of question and answer columns (Berant et al., 2013).

To enlarge the dataset, the OpenAI API and the GPT-3.5-turbo model were used to generate an additional 6,000 question-answer pairs.

After compiling the dataset with English questions and answers, translated versions of the questions and answers in Armenian were added using Google Sheets' Google Translate function.

## 2. Model Deployment and Response Generation

The initial phase of deploying the mGPT language model involved setting up the environment. This included installing PyTorch for a flexible computing framework and the HuggingFace library for loading and managing pre-trained models and their tokenizers. Specifically, the GPT2 Tokenizer and GPT2LMHeadModel from the transformers library were used to load the mGPT model and its tokenizer.

The response generation process was implemented through a function that takes a question as input and generates a corresponding answer. The core of this function lies in the use of the `model.generate` method, which produces text based on several parameters: min_length and max_new_tokens set boundaries on the length of the generated text, ensuring that responses are neither too brief nor excessively long. The use of eos_token_id and pad_token helps define the end of the sequence and padding behavior, respectively. Additionally, the top_k and top_p parameters control the randomness of the text generation, with top_k=10 making the model's choices among the 10 most likely next words. The no_repeat_ngram_size=5 prevents repetitive sequences of up to five words, promoting more diverse and engaging text.

The generated responses are then stored in a dataframe for further investigation and analysis.

## 3. In context learning

The first approach used to investigate and evaluate the model's abilities and quality was the zero-shot prompting strategy, where the model generated responses without seeing any prior examples. Given the limitations of our resources and time constraints, a sample of 30

question-answer pairs was selected and provided to the model, as generating responses required considerable time.

To evaluate the text similarity between the generated and reference answers, the multilingual BERT model (mBERT) was employed due to its ability to handle multiple languages, including Armenian, and can better capture the nuances and semantics of non-English texts (Devlin et al., 2018). Both the generated and reference answers were converted into BERT embeddings, and the cosine similarity between these vectors was calculated.

To expand the experiments, a few-shot prompting strategy was also employed to analyze whether in-context learning techniques could improve the model's performance using minimal resources. For few-shot prompting, examples were included with the prompt, and the generated responses were again compared with the reference responses.

The results of both zero-shot and few-shot prompting were not promising, as evidenced by the generated text and the calculated similarity scores. Therefore, it was decided to proceed with fine-tuning the model for the question-answering task. A detailed analysis of these outcomes is provided in the ‘Results’ section.

## 4. Fine-tuning

To fine-tune the mGPT model on an Armenian question-and-answer (Q&A) task, several critical steps were undertaken, encompassing data preprocessing, tokenization, dataset preparation, model initialization, and training configuration.
We began with a dataset stored in a DataFrame, which included columns for questions and answers. The initial step was to remove rows containing any NaN values in these columns to ensure data completeness. Next, a custom function, `clean_text`, was applied to strip out the words "question" and "answer" from the respective columns, standardizing the inputs and ensuring consistency. Following this, any rows where the questions or answers were empty after cleaning were filtered out, thus refining the dataset further.

### a. Tokenization

Tokenization was initially done using BPE (Byte Pair Encoding). Given that the tokenizer was not functioning correctly, we utilized mGPT’s tokenizer from the Hugging Face Transformers library as an alternative. A tokenization function was crafted to process the dataset efficiently. The questions were prefixed with "question: " and suffixed with "answer" to clearly delineate the expected answer format. Both the formatted inputs and answers were tokenized, with a maximum sequence length set to 128 tokens. Padding and truncation were applied as necessary.

Furthermore, pad tokens in the labels were replaced with -100 to ensure they were ignored during the loss calculation. This function was then applied in batches, and the original 'question' and 'answer' columns were removed post-tokenization.

b. Dataset Preparation

With the tokenized dataset ready, we proceeded to split it into training, validation, and test sets. An initial 80-20 split created the training and test subsets. The test subset was then split equally into validation and test sets. These splits were consolidated into a `DatasetDict` object, allowing streamlined access during training.

c. Model Initialization

Next, we initialized the mGPT model using the `AutoModelForCausalLM` class by loading it with parameters from the "ai-forever/mGPT-1.3B-armenian" model. As the model had 1.3B parameters and our GPU had 16GB RAM, we tried to perform quantization and loaded a 16 bit version of the model. Later, a much better GPU was purchased, so we switched back to the 32 bit version of the model to avoid information loss.

d. Training

The training process was meticulously configured using the `TrainingArguments` class. Key parameters included setting the output directory for saving model checkpoints and logs, evaluating the model at the end of each epoch, and fixing the learning rate at 5e-5. The per-device training batch size was set to 1, with gradient accumulation steps set to 64. We scheduled a total of 5 epochs and applied a weight decay rate of 0.01 to regularize the model. Logging was configured to record metrics every 50 steps, and checkpoints were saved at the end of each epoch. Mixed precision training was enabled, and the optimizer "adamw_8bit" was employed to perform memory-efficient optimization. Finally, a `Trainer` object was instantiated with the model, training arguments, and the tokenized datasets.

## Results

This section presents the findings from evaluation of the model's performance using in-context learning and fine-tuning.

### 1. In-Context Learning Results

For the in-context learning, the similarity between the reference answers and the generated responses was evaluated using the mBERT model. The results of the mBERT similarity scores

and zero-shot generated answers are presented in Table 1. In the few-shot setup, the questions provided in the prompt were repeated in the answers, resulting in very long responses. The mBERT similarity scores indicate the degree of semantic similarity between the reference and generated responses, with higher scores representing greater similarity. These scores provide a quantitative measure of the model's performance in generating contextually relevant responses in Armenian.

| **Question** | **Reference Answer** | **Generated Response ZS** | **mBERT Similarity Score** |
|---|---|---|---|
| Ո՞րն է մեր արեգակնային համակարգի ամենամեծ մոլորակը: | Յուպիտեր | Ո՞րն է մեր արեգակնային համակարգի ամենամեծ մոլորակը: Ինչպես տեսնում ենք, այն բավականին «բարձր է» երկրայի կողմից: Այ | 0.35 |

Table 1: Example of zero-shot response and its similarity score

The overall statistics for the zero-shot setups demonstrate a mean score of 0.71 with a standard deviation of 0.12. In comparison, the mean score for the few-shot setup decreases to 0.62, while the standard deviation remains relatively the same as in the zero-shot setup. This decrease in similarity score is unexpected, as providing several examples to the model is typically expected to generate more accurate responses based on the provided examples.

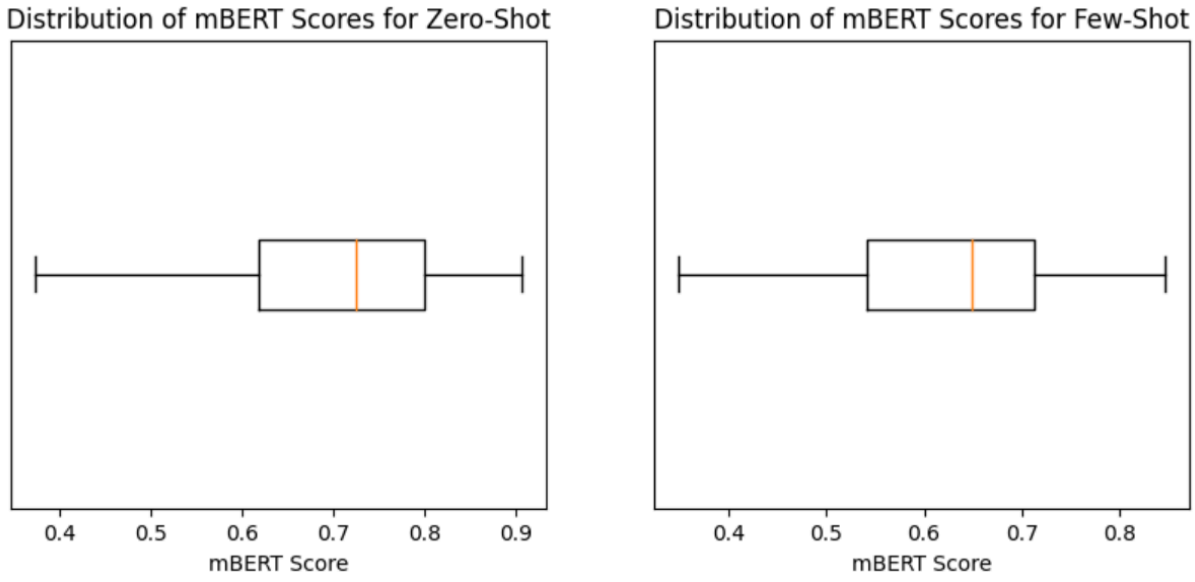

Figure 1: The distribution of mBERT scores for two different setups

It is important to note that several factors can impact mBERT similarity scores. One significant factor is the translation used to create the Armenian dataset. Since the translation was performed using Google Translate, which can be not accurate, poor translation quality may affect the similarity scores. The translation process can introduce similarities not present in the original text, leading to increased similarity scores. Another factor is surface-level similarity. Shared vocabulary or similar sentence structures can result in higher similarity scores, even when the true semantic meaning differs.

When comparing zero-shot results with few-shot results, the quality of few-shot examples, including their structure, length, and complexity, can impact the outcomes and potentially decrease the model's performance. To mitigate this issue, the prompt examples were simplified and carefully translated to ensure accuracy and avoid unnecessary complexity.

**2. Fine-tuning Results**

The fine-tuning process of the mGPT model on the Armenian Q&A task encountered several challenges and required careful adjustments to optimize performance.

Initially, the model was trained on a machine with 16GB of RAM. Due to memory constraints, the training process had to be conducted with reduced parameters, which often led to runtime disconnections or crashes. Additionally, misconfigurations in the training arguments sometimes resulted in training loss values of 0, caused by improper data processing or the generation of NaN values during tokenization.

To address these issues, the training environment was upgraded to a machine with an 80GB RAM GPU, significantly improving the stability and efficiency of the training process. The Hugging Face tokenizer provided by the mGPT model was used, as integrating a BPE tokenizer with mGPT proved to be challenging.

The training was carried out over five epochs, with the following epoch-wise loss metrics recorded

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 0 | 2.710500 | 2.690527 |
| 1 | 2.438100 | 2.585707 |
| 2 | 2.395100 | 2.554629 |

| 3 | 2.282600 | 2.554540 |
| 4 | 2.235700 | 2.596454 |

These results demonstrated a consistent reduction in both training and validation loss across epochs. Notably, there was a significant decrease from epoch 0 to epoch 1, indicating effective learning and model convergence. This trend continued through subsequent epochs, showcasing the model's ability to adapt and fine-tune effectively on the Armenian Q&A task.
Despite the extensive fine-tuning process, including data preprocessing, tokenization, and optimizing training parameters, the final implementation of the model faced challenges. After fine-tuning, we attempted to generate answer for the question in Table 2.

| **Before fine-tuning mGPT_armenian** | **After fine-tuning mGPT_armenian** |
|---|---|
| *Generated Answer:* "question: ինչպես է կոչվում Հարրի Փոթերի առաջին վեպի անունը: <answer> Որոշ հեղինակներ կարծում են, որ վեպը սկսվում է հենց Հարրի Փոթերի մասին գրած նամակով, որը նա ուղարկել էր Հոլմսին միայն վ�" | *Question asked:* "ով է իմ ֆարդաջյանը?" *Generated Answer*: Ղի������խՂ�������ղ�ա��ա�ի�ննկ��ե�ըա�� �" |

Table 2: Comparison before and after fine-tuning mGPT_armenian

However, the generated answer contained issues. This output indicates that there may be a problem with the tokenization process or another aspect of the model's configuration. The presence of question marks in the text suggests that the tokenizer might not be correctly handling the input or output sequences, potentially due to encoding issues or incompatibilities between the tokenizer and the model.

## Conclusion

In this paper, we presented a comprehensive study on fine-tuning the mGPT-1.3B-Armenian model for a question answering task. Through our experiments, we attempted both zero-shot and few-shot learning approaches, which provided initial insights into the capabilities of the pre-trained model. However, these initial tests highlighted the limitations in understanding and generating contextually relevant and semantically accurate Armenian language content. This led to the decision to proceed with fine-tuning the model.

Despite the successful reduction in training and validation loss, the fine-tuning process did not fully resolve all issues, particularly with the tokenization step. The generated answers contained non-readable characters, indicating a need for further investigation into the tokenization process and model compatibility. Future work should focus on addressing these tokenization challenges to ensure accurate and coherent answer generation.

**Individual Contributions**

Here are the specific contributions made by each team member to the project. While specific tasks were divided among members, the entire project was carried out collaboratively, with everyone helping each other and adapting tasks as needed to ensure the project's completion.

Astghik Kostanyan

- Responsibilities:
    - Coordinated overall project activities and managed the timeline.
    - Researched relevant models and performed model selection.
    - Fine-tuned the mGPT model for Q&A downstream task and handled the computational resources.
    - Wrote fine-tuning methodology and results in the paper.

Anahit Navoyan

- Responsibilities
    - Conducted data generation, data preprocessing and cleaning.
    - Led the implementation of the zero-shot and few-shot prompting strategies.
    - Implemented the mBERT similarity evaluation.
    - Wrote in-context methodology and results in the paper.

Anahit Zakaryan

- Responsibilities
    - Conducted literature review and identified relevant studies and models.
    - Conducted data generation, translation and preparation.
    - Conducted analysis.
    - Wrote data collection and model deployment methodology in the paper.

**References**

Shliazhko, O., Fenogenova, A., Tikhonova, M., Mikhailov, V., Kozlova, A., & Shavrina, T. (2022, April 15). *MGPT: Few-Shot learners go multilingual*. arXiv.org. https://arxiv.org/abs/2204.07580

Berant, J., Chou, A., Frostig, R., & Liang, P. (2013). Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1533-1544). Association for Computational Linguistics. https://www.aclweb.org/anthology/D13-1160

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018, October 11). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv.org. https://arxiv.org/abs/1810.04805