Quantum theory helper

Danil Kirnos

December 2024

Abstract

This project implements a model to answering questions related to quantum theory and quantum information systems. The main purpose is to help prepare for tests in university and being able to solve some of them. The dataset was collected from tests in the university course.
`https://github.com/bivba/prikols`

1 Introduction

When the quantum information systems course started at the university, i realized how difficult it is to understand the concepts of quantum theory and underlying math. We are learning wave functions, operators, and matrices, which is all fine when we follow the formulas, but it is hard to connect the math to real physical phenomena, especially in case of using them in real quantum algorithms. And when i was taking the tests, it was very difficult to find accurate information towards them.

1.1 Team

This project was prepared by Danil Kirnos. All of the work was done by myself.

2 Related Work

there are a few systems that imitates work of quantum circuits and simulates quantum computing process. One of the most popular product is [1] Qiskit. The name "Qiskit" is a general term referring to a collection of software for executing programs on quantum computers. Most notably among these software tools is the open-source Qiskit SDK, and the runtime environment (accessed using Qiskit Runtime) through which you can execute workloads on IBM® quantum processing units (QPUs). Using Qiskit it is possible to create circuits using python code.

Early question-answering systems used rule-based methods that relied heavily on handcrafted features.[2] Recurrent architectures such as LSTMs and GRUs improved the handling of sequential data. [3] Recently, Transformer-based models like BERT and GPT have set new benchmarks in natural language understanding tasks, including question answering. These models [4]leverage self-attention mechanisms and large-scale pretraining on diverse datasets, achieving state-of-the-art results in various domains.

3 Dataset

It is very hard to find accurate data with questions regarding quantum theory and detailed and truthful answers to them. Data collection was carried out manually by rewriting questions from previous year's tests into a latex code. In our possession were around 80 questions and thus all of them were used for training data. Questions vary from general theory questions to practical, where it is needed to compute multi-cubit circuit of gates.

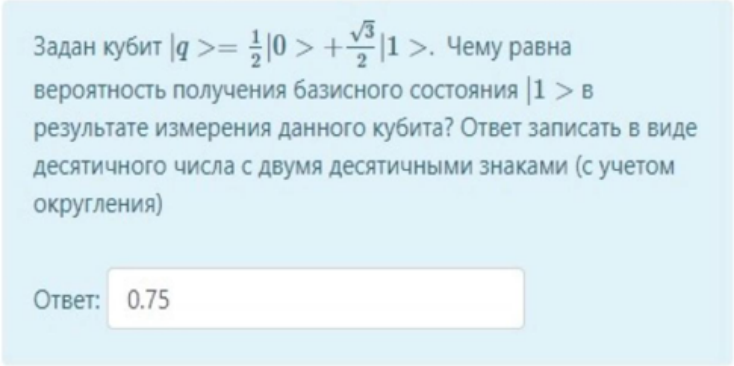

Figure 1: Question example

Figure 2: Question example

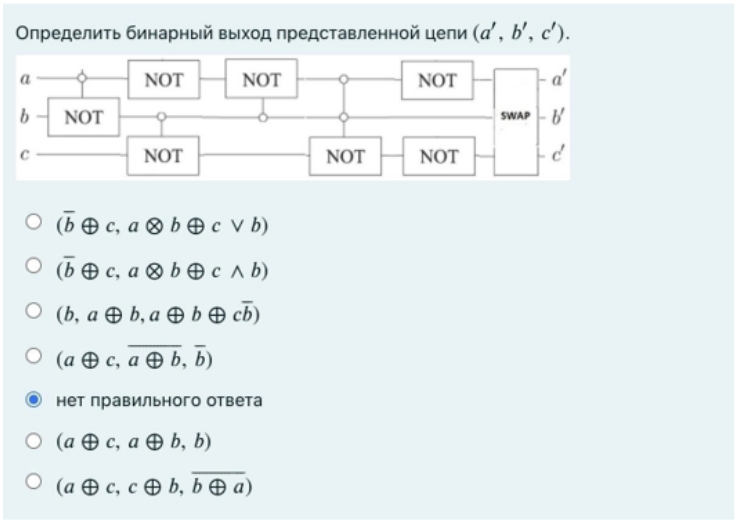

Figure 3: Question example

Also to enhance the dataset and provide better general understanding three datasets from huggingface with common questions were used.
`https://huggingface.co/datasets/jilp00/YouToks-Instruct-Quantum-Physics-I`,
`https://huggingface.co/datasets/jilp00/YouToks-Instruct-Quantum-Physics-II`,
`https://huggingface.co/datasets/jilp00/YouToks-Instruct-Quantum-Physics-III`.

Dataset was preprocessed for better model's understanding and more efficient answering.
To make the training process more effective was created meaningful prompt. This prompt was designed to clarify the intent of the question, ensuring that the model received well-defined inputs during training.

Also was collected corpus of theoretical data consisting of lecture notes, articles from `https://quantum-ods.github.io/qmlcourse/book/index.html`, and quantum computing book[5]. This corpus has 296.000 number of words and it was used for providing model context during training for better understanding of a topic. The most relevant part of the document was passed as additional context during training. All the text was translated to the english before passing to the model for better understanding as part of the text in corpus in Russian and part in English and all the questions were in Russian so it is important to uniform languages.

Figure 4 is showing one of the prompts for model after applying prompt template and providing additional context for the answer

Figure 4: Prompt example

Answers from the new test that was already passed were used to collect train dataset. There were around 20 unique questions that had not appeared in the train data.

4 Model Description

I decided to use two large language models: LLama3.1 and Qwen2.5. To provide model with context for better understanding was used FAISS by Meta. Faiss is built around an index type that stores a set of vectors, and provides a function to search in them with L2 and/or dot product vector comparison.

$$d(p,q) = ||p-q|| = \sqrt{\sum_{i}^{n}(p_i - q_i)^2} \tag{1}$$

equation (1) defines L2 distance.
Some index types are simple baselines, such as exact search. The optional GPU implementation provides what is likely the fastest exact and approximate nearest neighbor search implementation for high-dimensional vectors, Lloyd's k-means, and small k-selection algorithm [6].

All the models were pretrained and quantized to 4 bit precision by unsloth. Applying quantization to reduce the weights of a neural network down to a lower precision naturally gives rise to a drop in the performance of the model. This is commonly measured as a difference in perplexity between the original and quantized models on a dataset which is downstream task agnostic. Perplexity can be thought of as the model's uncertainty in predicting the next word in a sequence, it is the exponential negative log-likelihood of a sequence of tokens.

Minimizing this drop in performance while compressing an LLM to ever lower precision is a key challenge, and many new techniques have been proposed to reduce performance loss. When reducing the precision of a tensor, the range of values that can be represented will usually drop. For example, when reducing a 32-bit floating point down to an 8-bit integer, for example, the range will be lowered from [-3.4e38, 3.40e38] to [-127, 127]. This means that the tensor will need to be mapped from one range to another and rounded to the nearest value. This can be represented as:

$$X^{INT4} = round(c^{FP32} \cdot X^{FP32})$$

where c^{FP32} is the quantization constant. To train such large models one of the parameter-efficient training method was used: Low-Rank Adaptation (LoRa)[7].For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, we constrain its update by representing the latter with a low-rank de composition $W_0 + \Delta W = W_0 + BA$,where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$,and the rank $r \ll min(d, k)$. During training, W_0 is frozen and does not receive gradient updates, while A and B contain trainable parameters. The modified forward pass will be presented as:

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

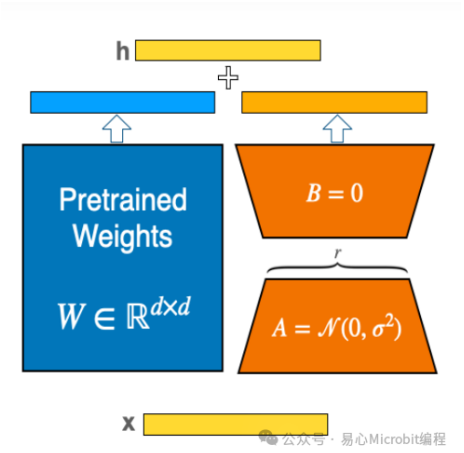

Figure 5: Lora structure

For training strategy Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO)[8]. The base model is fine-tuned on a supervised dataset of questions and their corresponding answers to provide a basic understanding of domain knowledge. After that DPO was used teach model answering questions correctly. DPO simplifies the RLHF process by eliminating the need for a separate reward model. Instead, it directly optimizes the model to align with preferences using preference probabilities.

5 Experiments

5.1 Metrics

As the main goal of the model is to properly answer the questions and solve tests, accuracy of solving tests was chosen as the evaluation metric.

$$\text{Accuracy} = \frac{\text{number of correctly chosen answers}}{\text{number of all answers}}$$

Here, if the question contains more than one correct answer, it is counted as correct only if all options are chosen. If model does not give exact answer as one of the choice, but it is hidden in the text, it is also counts as correct.
If the question requires numerical answer, it is considered correct if the output of the model is the exact same number even if it is without any calculations.

5.2 Experiment setup

5.2.1 FAISS

FAISS was used in prompt processing to store vectorized corpus for making context for each questions. Also it was used to find most similar chunks to each question. Hyperparameters:

- distance metric = L2 distance
- chunks length in context = 256
- context split overlap = 32
- number of most similar chunks (K) = 1

5.2.2 LLama3.1

The first model that was chosen is Llama3.1, that was quantized to 4 bit. Hyperparameters of the model are:

- max sequence length = 2048.
- precison = INT4.
- number of parameters = 7B.

- rank of the Low-rank decomposition (r) = 32.
- lora dropout = 0.
- lora alpha = 16.
- loftQ = None.
- use gradient checkpointing = unsloth.
- lora modules = query projection, key projection, value projection, output projection, up projection, down projection.

5.2.3 Qwen2.5

Second chosen model was Qwen2.5 base, also reduced to 4bit.
Its hyperparameters:

- max sequence length = 4096.
- precison = INT4.
- number of parameters = 14B.
- rank of the Low-rank decomposition (r) = 16.
- lora dropout = 0.
- lora alpha = 16.
- loftQ = None.
- use gradient checkpointing = unsloth.
- lora modules = query projection, key projection, value projection, output projection, up projection, down projection.

5.2.4 Training Strategy

Supervised Fine-Tuning (SFT):

The first training phase included SFT using a full dataset of question-and-answer pairs with general question. This step ensured that the model was equipped to understand the structure and context of typical question. After that model was trained directly on questions from university.

Hyperparameters of SFT training:

- train batch size = 4.
- gradient accumulation steps = 4.
- warmup steps = 5.
- number of epochs on full data = 0.33.
- number of epochs on partial data = 8.
- learning rate = 2e-4.
- optimizer = "AdamW 8bit"
- weight decay = 0.01
- lr scheduler = linear

DPO Training:
Upon the completion of SFT the dataset for Direct Preference Optimization (DPO) training was generated. This dataset consist of accepted and rejected answers of the model. Hyperparameters of DPO training:

- per device train batch size = 1.
- gradient accumulation steps = 4.
- warmup ratio = 0.1.
- number of train epochs = 6.
- learning rate = 5e-6.
- optimizer = AdamW 8bit.
- weight decay = 0.0.
- lr schedulertype = linear.
- max length = 512.
- max prompt length = 512.

6 Resluts

The initial experiment used the LLaMA model. After an epoch of training, model failed to provide meaningful answers to the questions. Instead, it mostly generated reformulations of the questions themselves, which rendered it unsuitable for the intended task. As a result, it was decided to switch to drop it from experiments.

After completing training for Qwen2.5 the following results on train data (Tab. 1) were obtained.

model	**solved questions**	**accuracy**
SFT'd model	25	0.33
gpt4o-mini	34	0.45
DPO model	35	0.47

Table 1: performance of the model on train data

And the results on the test data in Tab. 2 respectively

model	**solved questions**	**accuracy**
DPO model	13	0.62
gpt4o-mini	12	0.57

Table 2: performance of the model on test data

Model trained only using SFT was not evaluated on test data because of it's poor performance on train data.

7 Conclusion

The dataset was collected using solved tests from course in university. Model was trained using two approaches: Supervised fine-tuning and Direct Preference Optimization. With this i was able to achieve results on tests better than gpt4o-mini, especially on theoretical questions but it is struggles with calculations and mathematical problems.

References

1. Qiskit https://www.ibm.com/quantum/qiskit
2. Hochreiter, S., Schmidhuber, J. (1997). "Long Short-Term Memory."
3. Devlin, J., et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."
4. Vaswani, A., et al. (2017). "Attention Is All You Need."
5. Peter Y. lee, Huiwen Ji, Ran Cheng *Quantum Computing and Information*, 2024, Polaris QCI Publishing
6. Billion-scale similarity search with GPUs https://arxiv.org/abs/1702.08734
7. LoRA: Low-Rank Adaptation of Large Language Models https://arxiv.org/abs/2106.09685
8. Direct Preference Optimization: Your Language Model is Secretly a Reward Model https://arxiv.org/abs/2305.18290