Fine-Tuning Large Language Models for Multitasking in Online Shopping Using Synthetic Data

Innova team submission for Amazon KDD Cup 2024 Challenge for LLMs - 4th position in the Multilingual Track

Fernando Sebastián Huerta*
Innova-tsn
Madrid, Spain
fernando.sebastian@innova-tsn.com

Julián Rojo García*
Innova-tsn
Madrid, Spain
julian.rojo@innova-tsn.com

Carla Martín Monteso*
Innova-tsn
Madrid, Spain
carla.martin@innova-tsn.com

Roberto Lara Martín*
Innova-tsn
Barcelona, Spain
roberto.lara@innova-tsn.com

Leyre Sánchez Viñuela*
Innova-tsn
Madrid, Spain
leyre.sanchez@innova-tsn.com

Carlos de Leguina León*
Innova-tsn
Madrid, Spain
carlos.leguina@innova-tsn.com

Rubén Sordo López*
Innova-tsn
Madrid, Spain
ruben.sordo@innova-tsn.com

Abstract

This paper presents an approach by the Innova-team to the KDD Cup 2024 ShopBench challenge, specifically detailing the 4th-place solution in Track 4: Multi-lingual abilities [1]. The research introduces a versatile Large Language Model (LLM) based on Qwen2-72B-Instruct, designed to enhance the multi-lingual online shopping experience. Utilizing multi-task learning, the model was fine-tuned to address various tasks derived from Amazon shopping data. To optimize performance, the vLLM library [**vLLM**] was employed in conjunction with Activation-aware Weight Quantization (AWQ) [1], enabling efficient model inference across four NVIDIA T4 GPUs in the competition environment. This solution demonstrates the potential of LLMs in mastering complex multi-lingual e-commerce tasks, ranging from product navigation to personalized recommendations. The research leverages Qwen2-72B-Instruct [2] as the foundation for fine-tuning, showcasing its effectiveness in tackling multi-lingual e-commerce challenges.

The code and datasets are publicly available in the following GitLab repository:

https://gitlab.aicrowd.com/fersebasIn/innova-team-amazon-kdd-cup-2024-track-4-4th-position

Keywords

Large Language Models, KDD Cup, Qwen, AWQ, Fine-tuning, E-commerce, Multitask

ACM Reference Format:
Fernando Sebastián Huerta, Julián Rojo García, Carla Martín Monteso, Roberto Lara Martín, Leyre Sánchez Viñuela, Carlos de Leguina León, and Rubén Sordo López. 2024. Fine-Tuning Large Language Models for Multitasking in Online Shopping Using Synthetic Data: Innova team submission for Amazon KDD Cup 2024 Challenge for LLMs - 4th position in the Multilingual Track. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference ACM KDD 2024).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

E-commerce is a complex field involving multiple tasks, from browsing to purchasing, all of which require deep insights into customer behavior and intentions. This complexity necessitates multi-task learning models capable of leveraging shared knowledge across various aspects of the online shopping process.

However, many current models lack the **specific product knowledge** necessary for e-commerce tasks, often struggling to distinguish between similar products from different brands, understand compatibility between various accessories and devices, recognize the significance of product generations or versions, interpret technical specifications and their implications for use, and identify which features are standard versus optional for a given product category. This limitation hinders their ability to provide accurate recommendations, answer nuanced product-related queries, or make informed compatibility assessments, ultimately impacting the quality of the e-commerce experience they can support.

Therefore, to address these limitations, specialized fine-tuned models for e-commerce, known as e-commerce LLMs, have been developed, such as [3], [4], and [5]. In this competition, the challenge is to utilize an Open Large Language Model (LLM) to handle diverse tasks for which the model has not been specifically trained, as the test dataset remains hidden from the participants. This setup evaluates the model's ability to generalize its understanding of e-commerce concepts and apply its knowledge to novel

*These authors contributed equally to this research.
[1]Challenge Web

Conference ACM KDD 2024, August 25-29, 2024, Barcelona, Spain

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

situations, mimicking the real-world scenario where new products and queries constantly emerge.

1.1 Data exploration

The ShopBench Dataset is an anonymized, multi-task dataset derived from real Amazon shopping data. It is designed to evaluate four key shopping skills: Shopping Concept Understanding, Shopping Knowledge Reasoning, User Behavior Alignment, and Multi-lingual Abilities, which correspond to Tracks 1 through 4 of the Amazon KDD Cup, respectively.

The focus of this paper is on the Multi-lingual Track, which includes multi-lingual concept understanding and user behavior alignment. Statistics for ShopBench related to this track are provided in Table 1:

Table 1: Dataset statistics for Track 4: Multi-lingual Abilities.

#Tasks	#Questions	#Products	#Queries
7	2379	~6000	~520

Note: Product categories, attributes and reviews do not appear in the statistics for this track.

In addition, the dataset is divided into a few-shot development set and a test set to better replicate a few-shot learning scenario. The development dataset[2] is provided in JSON format and includes the following fields:

- `input_field`: The instructions and the question that the model needs to answer.
- `output_field`: The correct answer to the question.
- `task_type`: The type of task, which can be Multiple Choice, Retrieval, Ranking, Named Entity Recognition (NER), or Generation.
- `metric`: The metric used to evaluate the answer.

Listing 1: Example of JSON code.

```
{"input_field":"Which of the following productcategories may have
    the attribute power source? \n0. table \n1. writing tools \n2
    . car seat cover \n3. comb \nAnswer:", "output_field":3, "
    task_name":"task2", "task_type": "multiple-choice", "metric
    ":"accuracy", "is_multiple_choice": true, "track": "amazon-
    kdd-cup-24-understanding-shopping-concepts"}
```

However, the test dataset follows a different format with only two fields:

- `input_field`: Identical to the one in the development set.
- `is_multiple_choice`: Indicates whether the question is a multiple-choice question, with a value of 'True' or 'False'.

Additionally, in order to obtain a more elaborated training set, examples have been generated for several tasks such as the following:

- **Task 1 (Generation)**: Explanation of categories or product types.
- **Task 4 (NER)**: Named Entity Recognition in queries.
- **Task 5 (Multiple-choice)**: Questions based on product descriptions.

1.2 Data evaluation

Since there are various types of tasks, the evaluation method for each of them differs:

[2]URL development dataset

Table 2: Evaluation metrics.

Task Type	Metric
Multiple Choice	Accuracy
Retrieval	Normalized Discounted Cumulative Gain
Ranking	Micro-F1 score
Named Entity Recognition	Hit@3
Generation	- ROUGE-L for Extraction tasks [6]
	- BLEU score for Translation tasks [7]
	- Sentence Transformers (cosine similarity)

Since all these metrics fall within the range [0, 1], the average metric is calculated across all tasks within each track using macro-averaging to determine the overall score for that track.

Note that in this multilingual Track 4, out of the five task types, there are no Named Entity Recognition or Retrieval tasks.

2 Detailed Method

The solution proposed for this challenge involves several steps. First, a larger synthetic dataset was created. The base model was then fine-tuned with this dataset to enhance its performance. Following that, the model was optimized, and its efficiency was improved through padding techniques and model quantization strategies. The inference process leveraged the vLLM library for optimized parallelization across multiple GPUs, ensuring efficient handling of both multiple-choice and open-ended tasks. Additionally, prompt engineering played a crucial role in enhancing the model's responses, particularly by crafting task-specific prompts that aligned with the nature of each question, significantly improving the model's performance in ranking (output format) and generation tasks (maximum tokens).

Although the paper focuses on the solution for Track 4, most of the steps were replicated for the other tracks, resulting in the following positions: Track 1—6th, Track 2—6th, and Track 5—8th.

2.1 Data Creation

The original dataset available for the participants of the competition (the development dataset) consisted of only 96 items, and its structure was presented in the Subsection 1.1.

To enrich the collection of examples, different public datasets and an OpenAI model were used, in particular GPT-3.5. The new synthetic examples covered tasks from 1 to 11, with the exception of tasks 9 and 10.

2.1.1 AMAZON PRODUCTS DATASET. One of the main external resources was the Amazon-M2 dataset [8] which included a list of Amazon products with the following attributes: `id`, `locale`, `title`, `price`, `brand`, `color`, `size`, `model`, `material`, `author`, and `desc` (description).

The synthetic examples followed a similar structure to the development set, but only English-language products were used with a simpler format consisting of two columns: `input_field` and `output_field` .

The procedure involved defining a different prompt for each task, which was fed with the examples from the corresponding task found in the development set, and a list of Amazon products. At least 2,500 products were used for each task, and five examples were generated for each product. Afterwards, the resulting examples had to go through a cleaning process that removed the ones that did not match the structure of the `input_field` and `output_field` attributes of the original examples.

2.1.2 OTHER DATASETS. Another external dataset containing a list of Amazon product categories was used in task 1, as its purpose was to explain categories and types of products. Meanwhile, for tasks 6 and 7 a dataset of Amazon reviews [9] was applied, since they involved keyphrase and aspect extraction from user reviews. The full dataset can be found in the folder "finetuning_code" of the repository.

2.1.3 COMPLETE TRAINING DATASET. By following all these methods, a minimum of 8,000 examples were produced for several known tasks. Moreover, to assemble the final training set, some data from the ECInstruct dataset was added. This is the first open-sourced, large-scale, and high-quality benchmark instruction dataset for e-commerce, and it was used for training the model eCeLLM [3] with 92,022 items.

Therefore, as the final step in creating the synthetic dataset, the generated examples were transformed into the ECInstruct format, which includes:

- `split`: Indicates whether the item will be used to "train" or "test" the model.
- `task`: The type of task (multiple-choice, generation, NER, etc.).
- `setting`: Always set to "IND_Single_Instruction".
- `instruction`: A sentence that specifies the action that needs to be carried out.
- `input`: The additional queries required in each instruction, for example, a product or a question.
- `options`: The list of options (if it is a multiple choice task).
- `output`: The answer (one or more numbers, or a string).
- `few_shot_example`: Always set to "null".

Note: In order to be able to use the setting "IND_Single_Instruction" and, therefore, simplify the instruction, the same sentence was always used within the examples of each task. For example: "Given the product title and question, select the correct answer among all the options."

Only the columns `input`, `options`, `instruction` and `output` of the "train" split were kept, so the final dataset used for training contained 204,593 items.

An example of the process is shown in Appendix A.

2.2 Fine-tuning Model

As mentioned previously, the base model used for instruction fine-tuning was the Qwen2-72B-instruct (72 billion parameters). Specifically, to reduce costs, the fine-tuning was performed on its 4-bit version, available on Hugging Face[3].

For fine-tuning, due to its size, the unsloth library[4] was used to reduce the required VRAM size.

The environment used was a **RUNPOD** with an RTX 6000 ADA GPU (48 GB of VRAM). The fine-tuning process involved 22,000 steps, equivalent to approximately 70 hours of execution time, at an estimated cost of $70.

Table 3 shows the specified fine-tuning parameters:

Table 3: Fine-tuning parameters.

Parameter	Value
Train batch size	2
Gradient accumulation steps	4
Max Steps	22000
Learning rate	2e-4
optim	Adamw 8bit
weight decay	0.01
lr scheduler type	linear

After fine-tuning, the model was saved in 16-bit precision to perform Activation-aware Weight Quantization (AWQ) instead of 4-bit quantization. More details can be found in the notebook inside the "finetuning_code" folder of the repository.

[3]Qwen Model unsloth 4bnb Hugging Face
[4]Unsloth GitHub

2.3 Padding Model

Before performing AWQ [1], it was necessary to adjust the model's parameter of `intermediate_size` from its default value of 29,568 to 29,696 in order to make inference using 4 GPUs.

The expansion of the model tensors from 29,568 to 29,696 significantly improved its compatibility with various GPU configurations for parallel processing. By increasing the size to 29,696, when it is divided by 128, 29,696 yields 232, which factors as $2^3 \cdot 29$. This factorization allows the model to work efficiently with 1, 2, 4, 8, or their combinations. In contrast, 29,568 divided by 128 gives 231, which factors as $3 \cdot 7 \cdot 11$. This less flexible factorization limits efficient parallelism to configurations with 1, 3, 7, or 11 GPUs, or their combinations.

The code for padding was found in an issue of the Qwen2 repository[5]. After executing the code, one must change the `intermediate_size` in config.json to the new value of 29,696 and copy the other files, excluding the weights. This change allows the use of the `vLLM` library across all 4 GPUs with `group_size` equals to 128 during inference.

2.4 Quantization Model

After evaluating various quantization techniques including GPTQ [10], AWQ, and Bits and Bytes [6], AWQ was chosen due to its flexibility, allowing different quantization levels based on `group_size`, and its compatibility with the `vLLM` library.

For this purpose, the `auto-AWQ` library by Casper-Hansen was utilized with the following quantization configuration:

Table 4: Quantization configuration.

Parameter	Value
zero_point	True
q_group_size	128
w_bit	4
version	GEMM

The size of the model after AWQ quantization was approximately 38.74 GB. More details can be found in the notebook inside the "compression_code" folder of the repository.

2.5 Inference/Serving Model Using vLLM

To ensure reproducibility, the model was initialized with a fixed random seed and utilized vLLM for efficient inference. Key configuration parameters included the batch size (16), tensor parallel size (4), and GPU memory utilization (1, i.e., 100%), all of which can be adjusted according to available computational resources.

The model utilized vLLM with half-precision computation, as bfloat16 is not supported on NVIDIA T4 GPUs. It employed worker distribution via Ray, and a maximum model length of 4,096 tokens was set to prevent potential RAM errors. The implementation requires pre-downloaded model weights for offline use.

The core of the implementation relied on its batch prediction method and baseline code, which could distinctly handle both multiple-choice and open-ended tasks, each with its own task-specific parameters.

Table 5: Task-Specific Parameters.

Parameter	Multiple-Choice task	No Multiple-Choice task
max_new_tokens	1	80
top_p	0.95	0.95
temperature	0.2	0
top_k	50	default value (-1)

[5]Qwen2 repository
[6]Bits and Bytes documentation

Every query was prefixed with the following system prompt: "You are a helpful online shopping assistant. Please answer the following question about online shopping and follow the given instructions.". Additionally, for non-multiple-choice tasks, a second instruction was appended: "Do not give explanation only Answer. Output:".

3 Experiments

A detailed summary of the most relevant tests conducted by the team, including their scores and a descriptions of the changes made, is provided below.

Table 6: Score of Selected Submissions.

Model	Score	Multiple-Choice Score	Generation Score	Ranking Score
Llama 3 70B Instruct	**0.669**	0.770	0.389	0.822
Qwen2 72B Instruct (no prompting engineering)	**0.696**	0.811	0.424	0.783
Qwen2 72B Instruct (with prompting engineering)	**0.711**	0.812	0.454	0.816

As shown in Table 6, the Qwen2 72B model consistently outperformed the Llama 3 70B across several key metrics and task types. Notably, the application of prompt engineering techniques significantly enhanced performance in the ranking task type, improving the score from 0.783 to 0.816. This result underscored the effectiveness of prompt engineering in optimizing model performance for specific tasks, making Qwen2 72B a better choice than Llama 3 70B in this competition.

4 Conclusions

The selection of the base model with the highest possible score that could fit into RAM in the execution environment was key, as was the **quality of the data used** to further enhance the base model through costly fine-tuning.

Another crucial aspect of the competition was recognizing that in most Tracks, the number of multiple-choice tasks was greater, thereby significantly influencing the final score. For this reason, efforts were directed at improving multiple choice without worsening the performance on other tasks, as it was already identified whether the task was multiple choice.

It's also worth noting that the parsing process was crucial, as there were instances where the LLM provided correct responses, but they were not formatted properly in the required output format. In summary, the success in achieving this position was largely due to the selection of the base model, the quality of the data, and the effectiveness of prompt engineering.

A Example Generation Task 5

Task 5 we named "Questions Based on Product Descriptions", and belongs to the Multiple-Choice type. Given a product name, a question, and a list of multiple possible answers (only one answer is correct), the model has to select the correct one.

A synthetic example created with a product from the Amazon-M2 dataset is presented below.

First, a specific prompt was provided to the GPT-3.5 model, which also included the product name and a few examples from the development set:

Listing 2: GPT-3.5 Prompt for Task 5.

```
{ "You are given a list of Amazon products that appear on an e-
      commerce website, and a question about it with 3 possible
      responses. Only 1 is correct and you need to indicate which
      one is it. Examples: {dev_data}.
Use the description from the following Amazon product to generate
      5 questions about its characteristics like in the previous
      examples. For each question provide 3 possible answers (2
      incorrect and 1 correct), and then indicate the answer (that
      is, the correct one).
Follow the same formulation as in the examples, and that is "The
      product '[HERE INSERT FULL PRODUCT NAME]' appears on an e-
      commerce website. [HERE INSERT QUESTION] [HERE INSERT OPTIONS
      ] Answer: [HERE INSERT NUMBER OF ANSWER]".
The possible answers must be numerated from 1 to 3 and the final
      answer must just be the number of the correct one.
Product description: {product}"}
```

The resulting output then had to go through a filtering process to ensure it had the correct structure, like the following:

Listing 3: Example of GPT-3.5 output for Task 5.

```
{ "input_field": "The product 'Microsoft PN7-00013 Bluetooth
      Mobile Mouse 3600 - Red' appears on an e-commerce website.
      What is the type of connectivity used in the mouse?\n1. USB\
      n2. Bluetooth\n3. Wi-Fi\nAnswer: ", "output_field": 2 }
```

Afterwards, it was transformed into the ECInstruct format:

Listing 4: Example in ECInstruct format for Task 5.

```
 { "split":"train", "task":"multiple-choice",
 "setting":"IND_Single_Instruction",
 "instruction":"Given the product title and   question, select the
       correct answer among all the options.",
 "input":{ "product title":"Microsoft PN7-00013
    Bluetooth Mobile Mouse 3600 - Red",
    "question":"What is the type of connectivity used in the mouse
         ?" },
 "options":"[[1. USB, 2. Bluetooth, 3. Wi-Fi]",
 "output":"2", "few_shot_example":null }
```

Finally, only the 'input', 'options', 'instruction', and 'output' fields were kept for training.

References

[1] Ji Lin et al. *AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration.* 2024. arXiv: 2306.00978 [cs.CL]. URL: https://arxiv.org/abs/2306.00978.

[2] An Yang et al. *Qwen2 Technical Report.* 2024. arXiv: 2407.10671 [cs.CL]. URL: https://arxiv.org/abs/2407.10671.

[3] Bo Peng et al. *eCeLLM: Generalizing Large Language Models for E-commerce from Large-scale, High-quality Instruction Data.* 2024. arXiv: 2402.08831 [cs.CL]. URL: https://arxiv.org/abs/2402.08831.

[4] Kaize Shi et al. *LLaMA-E: Empowering E-commerce Authoring with Object-Interleaved Instruction Following.* 2024. arXiv: 2308.04913 [cs.CL]. URL: https://arxiv.org/abs/2308.04913.

[5] Yangning Li et al. *EcomGPT: Instruction-tuning Large Language Models with Chain-of-Task Tasks for E-commerce.* 2023. arXiv: 2308.06966 [cs.CL]. URL: https://arxiv.org/abs/2308.06966.

[6] Chin-Yew Lin and Franz Josef Och. "Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics". In: *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04).* Barcelona, Spain, July 2004, pp. 605–612. DOI: 10.3115/1218955.1219032. URL: https://aclanthology.org/P04-1077.

[7] Kishore Papineni et al. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics.* ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135.

[8] Wei Jin et al. "Amazon-M2: A Multilingual Multi-locale Shopping Session Dataset for Recommendation and Text Generation". In: *arXiv preprint arXiv:2307.09688* (2023).

[9] Yupeng Hou et al. *Bridging Language and Items for Retrieval and Recommendation.* 2024. arXiv: 2403.03952 [cs.IR]. URL: https://arxiv.org/abs/2403.03952.

[10] Elias Frantar et al. *GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers.* 2023. arXiv: 2210.17323 [cs.LG]. URL: https://arxiv.org/abs/2210.17323.