

Fine-Tuning Large Language Models for Text-Based Recommendation

Adhrit Srivastav

Hemangani Nagarajan

Ishan Bhardwaj

Kien To

adhritsrivas@umass.edu hemanganinag@umass.edu ibhardwaj@umass.edu ktto@umass.edu

Abstract

Effective recommendation systems are essential for enhancing user experiences on online platforms. Text-based sequential recommendation systems, in particular, face the unique challenge of understanding user intent from textual interactions. Large Language Models (LLMs) present a promising solution, leveraging contextual information to provide personalized suggestions. This paper explores the fine-tuning of LLMs for text-based recommendations, with the aim of improving recommendation accuracy and user satisfaction.

Our methodology involves isolating reviewer interactions by considering reviewer IDs and the corresponding product ASINs associated with their purchases. We set a standard purchase history list size for each reviewer and conceal the remaining purchased products from the LLM. The model is then tasked with predicting a subsequent product of interest for the reviewer, aiming to assign the highest probability to the correct option out of the concealed products. This approach enhances the LLM's ability to understand and predict user preferences while not burdening it with predicting miniscule and irrelevant details, ultimately improving the effectiveness of text-based recommendation systems. Extensive experiments on a real-world dataset demonstrate significant improvements achieved by our method over existing approaches.

1 Introduction

The paper explores the potential of fine-tuning LLMs specifically for text-based recommendations, with the goal of improving accuracy and user satisfaction. It introduces a novel prompt mechanism that transforms relationship information into natural language text, which can then be processed by the LLM. The prompt mechanism we used allows the model to interpret user interactions and make more informed recommendations.

The project had a clear focus: to demonstrate the efficiency of fine-tuning LLMs for a specific task and using a diverse and expansive dataset. The first phase involved extracting and preprocessing review data. Subsets of data were extracted and converted into DataFrames, unnecessary columns were removed, and users with insufficient purchase histories were filtered out. The remaining data was then transformed into prompts, which formed the input for the LLM.

We explored various prompting techniques before settling on a three-part structure for each prompt: an instruction, input details, and the ground truth output. This format provided the model with a clear task, relevant information, and the desired output format.

To implement the model, the Unsloth AI and HuggingFace libraries were utilized, offering a range of pre-trained models and tools for customization. The option to apply QLoRA, a technique for improving model performance, was also considered. Consolidated SFTTrainer and TrainingArguments objects were created to centralize the process of changing hyperparameters, making it easier to adjust hyperparameters during training.

Initially, we attempted to work with Mistral-7B, a large-scale LLM. However, the computational cost required proved to be a significant roadblock. As a result, we scaled down to TinyLlama, a model with roughly 1/7th the number of parameters, which offered a more feasible option while still providing substantial performance.

2 Proposed vs. Accomplished Objectives

- ~~Collect and preprocess the dataset, focusing on essential categories and users with at least 5 products in their history, to mine product-customer relationships and construct contextual prompts.~~

- Expand upon pre-training in an already pre-trained LLM with a combination of subsets from the review dataset before fine-tuning. This, however, was not feasible as it was computationally very expensive.
- Fine-tune a next-word prediction LLM using the latest model and examine its performance. We did not proceed with this idea as it required a larger context size to send enough information into the model for the next product recommendation, necessitating a bigger model. When tested with a subset of the data, the compute was very expensive, leading us to adopt an alternative approach, which is explained later.
- Incorporate item-item and item-category relationships into prompts and train the model on these prompts to evaluate performance. We did not proceed with this as we switched to a smaller model with a smaller context size. Additionally, testing this approach on the larger model did not yield significant improvements.
- ~~Engineer prompts and outputs, and fine-tune the LLM to predict the correct output.~~
- ~~Optimize hyperparameters of the model trained on best-performing prompts to recommend the correct next purchase product for the customer.~~

3 Related work

The advent of neural networks brought significant improvements. (He et al., 2017) introduced Neural Collaborative Filtering (NCF), which used multi-layer perceptrons to model complex user-item interactions, significantly outperforming traditional methods. Similarly, (Xue et al., 2017) demonstrated the effectiveness of deep learning with their Deep Matrix Factorization model. Our approach builds on these advancements by leveraging the capabilities of LLMs to process and understand textual data, enhancing the personalization and accuracy of recommendations beyond traditional neural models.

The integration of LLMs like BERT (Devlin et al., 2018) and GPT-3 (Brown et al., 2020) has revolutionized natural language processing and brought significant advancements to recommender systems. (Geng et al., 2023) introduced the P5

framework, which unifies various recommendation tasks into a text-to-text paradigm, leveraging LLMs for comprehensive personalization. Similarly, (Lin et al., 2024) explored the integration of LLMs into different stages of the recommendation pipeline, enhancing feature engineering and user interaction. Our work builds on these advancements by specifically fine-tuning LLMs for text-based recommendation, utilizing a masking technique to focus the model’s attention on predicting concealed products based on user interactions. This method enhances the model’s ability to understand and generate relevant recommendations, addressing the unique challenges posed by text-based interactions.

Conversational recommender systems (CRS) have gained prominence for their interactive and explainable recommendations. (Gao et al., 2023) proposed Chat-Rec, a novel paradigm that utilizes ChatGPT to convert user profiles and interactions into prompts, enabling in-context learning and enhancing the interactivity of recommendations. This approach addresses key challenges such as the cold-start problem and cross-domain recommendations. Our methodology parallels this by leveraging the conversational capabilities of LLMs to understand and predict user preferences from textual data. By isolating reviewer interactions and employing a masking technique, we enhance the LLM’s ability to make accurate predictions, thereby improving recommendation quality and user satisfaction.

Zero-shot and few-shot learning facilitated by LLMs have also shown significant potential in recommender systems. (Sileo et al., 2021) demonstrated the effectiveness of LLMs in zero-shot recommendation by constructing textual prompts to estimate user-item affinities, bypassing the need for structured training data. Similarly, (Yuanxing Liu, 2023) explored the potential of LLMs in CRS, highlighting their ability to understand and control complex conversations. Our approach differentiates itself by fine-tuning LLMs specifically for text-based recommendations, addressing the unique challenges posed by textual interactions. By isolating reviewer interactions and employing a masking technique, we enhance the model’s ability to understand and predict user preferences accurately, even with limited data.

4 Dataset

4.1 Dataset description

For training and experiments, we sampled from the 2018 Amazon Review Dataset. This dataset consists of 233.1 million reviews between May 1996 and October 2018. The Amazon Review Data consists of two primary files: the dataset file containing reviewer ID, product title, product ID, review text, and reviewer rating, and the metadata file containing product details such as description, category, and brand. The data entries are stored in JSON format and can hence be easily converted into either Python dictionaries or Pandas data frames.

4.2 Data preprocessing

To prepare our dataset for training, we performed several preprocessing steps. Using NLTK, we downloaded a list of English words to filter out non-English words from the dataset later.

The main dataset contained review information, while the metadata contained product information. To ensure the reliability of the data, we filtered the dataset to include only verified reviews.

Next, we dropped columns that were not required for our analysis. From the review dataset, we removed columns such as overall rating, review time, and reviewer name. From the metadata, we removed technical specifications and image URLs. We then filtered the dataset to include only reviewers who had reviewed at least five products, ensuring that we had enough data for each reviewer to make meaningful predictions.

After resetting the index, we merged the review and metadata DataFrames on the product ASIN. We cleaned the text data in the 'title' column by removing special characters, digits, non-English words, and extra whitespaces.

We then grouped the data by reviewer ID, creating a nested structure with the number of products reviewed and their details. We filtered this grouped DataFrame to ensure it included only those reviewers with at least five products reviewed. New columns for prompts and outputs were created, where prompts described the products purchased by a reviewer and outputs indicated the next likely purchase.

Prompts were structured to initiate with a statement indicating the reviewer's purchase history, followed by details of the first four items associated with the reviewer. The prompts were de-

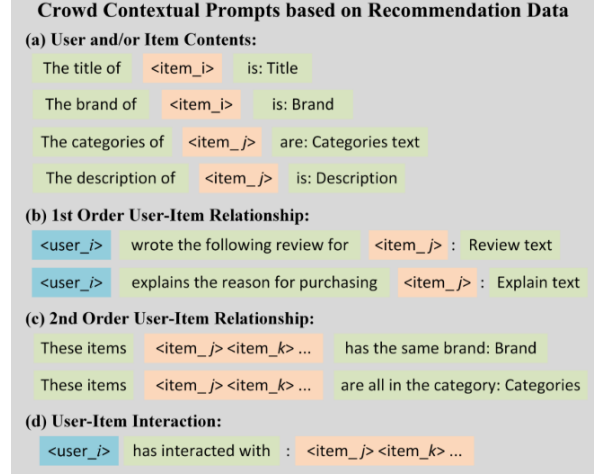


Figure 1: Crowd Contextual Prompts on Recommendation Data

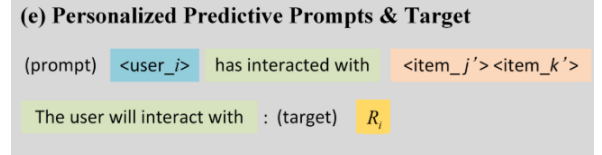


Figure 2: Personalized Predictive Prompts and Target

signed using the Crowd Contextual Prompts in Figure 1 (Xinyuan Wang, 2024) (Wang et al., 2024). They included a phrase contextualizing the purchase history, followed by four options for the next potential purchase. This was designed using the Personalized Predictive Prompts and Targets from Figure 2. Among these options, one represented the actual next purchase linked to the reviewer, while the remaining three were randomly chosen items not previously linked to the reviewer. An example of a prompt in the dataset is seen in Figure 3. The inclusion of randomized options and shuffling mitigated model bias and ensured robust recommendation generation.

To handle missing data, we filled in any missing titles with additional product titles from the metadata. The final DataFrame, containing only the prompts and outputs columns, was saved to a CSV file for training the recommendation model. This streamlined preprocessing pipeline ensured our data was clean, well-structured, and ready for training an effective recommendation system using large language models.

5 Baselines

We initially considered working with GPT-2 as our pre-trained model but the context size was quite

```

[{"prompts": "Reviewer has bought Starter Kit Unscented Count May Vary, Always Protection Daily Extra Count, Condensed Tomato pack, Volume Waterproof Black Brown Ounce May Vary Volume Mascara with Brush. Considering their purchase history, which product will they buy next from the below options? A) Duck quot Crystal, B) Brookside Dark Chocolate Goji and Raspberry Flavors Candy, 7 Ounce, C) Nestle Assorted Minis Bag, 19.75 Ounce, D) BSN Isoburn Protein Powder, Strawberry, 1.32 Pound.", "outputs": "Duck quot Crystal", "text": "Below is an instruction that describes a user's interaction history with products on Amazon. Write a response that appropriately completes the request.\n\n### Instruction:\nReviewer has bought Starter Kit Unscented Count May Vary, Always Protection Daily Extra Count, Condensed Tomato pack, Volume Waterproof Black Brown Ounce May Vary Volume Mascara with Brush. Considering their purchase history, which product will they buy next from the below options? A) Duck quot Crystal, B) Brookside Dark Chocolate Goji and Raspberry Flavors Candy, 7 Ounce, C) Nestle Assorted Minis Bag, 19.75 Ounce, D) BSN Isoburn Protein Powder, Strawberry, 1.32 Pound.\n\n### Response:\nDuck quot Crystal</s>"}]

```

Figure 3: Prompt Example

small (1024 tokens) given the amount of information we were providing in each prompt. Hence, for the baseline approach, we fine-tuned a Mistral-7B next-word prediction model on prompts containing the title, categories, and brand for each product a reviewer has purchased. The history size was set to a standard 4 products per reviewer, leaving at least one product that could be masked as a ground-truth target prediction for training. Our initial goal was to make the model predict the exact ground truth outputs without providing any options to choose from in the prompt. However, after training this model on a subset of 1523 prompts, we found that the model oftentimes did not *grammatically* predict the correct target prediction. This was because each target was a very specific string containing details like software version and copyrights. The predictions made were logically in the same domain as the ground truth prediction.

We generally divided the dataset into a 80-10-10 or 70-15-15 split for the training, validation, and test sets respectively. Before running any full passes through the training set, we tried to make our models overfit on a small subset of the training set, which we were able to accomplish. The main hyperparameters we considered were epochs, batch size, learning rate, and regularization (weight decay). We ran multiple tests with values of 5, 8, 0.0002, and 0.01 for these parameters respectively. Given the computational constraints for training, we started with these generally acceptable values and made slight tweaks based on the loss reduction pattern. We normally observed that the loss would show an initial reduction during training and then oscillate.

We chose to work with larger models for this project because we wanted the performance difference after fine-tuning to be very clear. We aimed to show that the results from the fine-tuned model were better than simply prompting the pre-trained model and that the reason for this difference could only be attributed to our fine-tuning process.

6 Our approach

Given our baseline results, we decided to use a scaled-down LLM to allow for more experiments and change our prompting strategy. We decided to work with TinyLlama, which has 1.1 billion parameters and can be coupled with QLoRA. This provided us with more opportunities for tuning and loss analysis while still providing comparable base performance to Mistral-7B. To get around the grammatical precision issue, we decided to provide the list of concealed products as options for the model to choose from in the prompt. The product that was assigned the highest prediction probability would be compared to the ground truth target product. This strategy maintained the essence of our original goal while not burdening the model with irrelevant details.

6.1 Unsloth

Unsloth is a powerful library designed to simplify and accelerate the training of LLMs. It integrates advanced techniques such as 4-bit quantization (for QLoRA) and gradient checkpointing, which significantly reduce memory usage and computational overhead. This allows us to train models with very long and changeable contexts, making them capable of handling extensive interaction histories from users. Unsloth's optimization strategies enable us to fit larger batch sizes into GPU memory, thereby speeding up the training process without compromising model performance.

6.2 LoRA and QLoRA

LoRA (Low-Rank Adaptation) (Edward J. Hu, 2021) is a technique that updates only a small subset of the model parameters, specifically targeting the attention layers. By adding low-rank matrices to these layers, LoRA reduces the number of parameters that need to be updated during training. This leads to faster training times and lower memory requirements while still achieving high performance. In our implementation, we used LoRA to update between 1% to 10% of the model parameters, significantly reducing the computational burden while maintaining the model's ability to learn from data effectively. Our model also uses 4-bit quantization to reduce memory usage. This is known as QLoRA (Tim Dettmers, 2023). The model is loaded with a specified data type (float16) suitable for the GPU available (Tesla T4, V100,

or Ampere). The use of 4-bit quantization allows us to significantly reduce the memory footprint of the model, enabling efficient training and inference even on hardware with limited memory.

6.3 Tinkering with data

Different prompt configurations were experimented with, including variations in prompt size and inclusion of product information such as title, brand, category, description, and reviews. Experimentation with Mistral-7B and TinyLlama models revealed constraints in training larger models due to computational limitations. Ultimately, the TinyLlama model, featuring only product titles in the prompt, demonstrated promising results despite its smaller size and training data. While the Mistral model showcased superior performance, its computational demands rendered it impractical for comprehensive dataset training. The TinyLlama model's smaller size and reduced resource requirements made it a more viable option. Additionally, the size of the training data influenced the model selection process. Despite its smaller size, the TinyLlama model proved effective when trained on the available dataset, highlighting the importance of leveraging available resources efficiently.

7 Model Architecture

As mentioned, we utilized Mistral-7B and TinyLLAMA, available through the Unsloth AI platform. We chose these models for their ability to handle large-scale data and perform complex language processing tasks efficiently. Both consist of an amalgamation of transformer blocks and feed-forward layers in their architectures.

7.1 Mistral-7B

Mistral-7B (Jiang et al., 2023) is a state-of-the-art LLM that excels in various natural language understanding and generation tasks. It is designed to manage extensive contextual information, making it well-suited for sequential recommendation tasks where understanding user interaction history is crucial. We leveraged the capabilities of Mistral-7B to process user purchase histories and generate personalized product recommendations.

7.2 TinyLLAMA

TinyLLAMA (Zhang et al., 2024) is a compact yet powerful LLM optimized for performance and

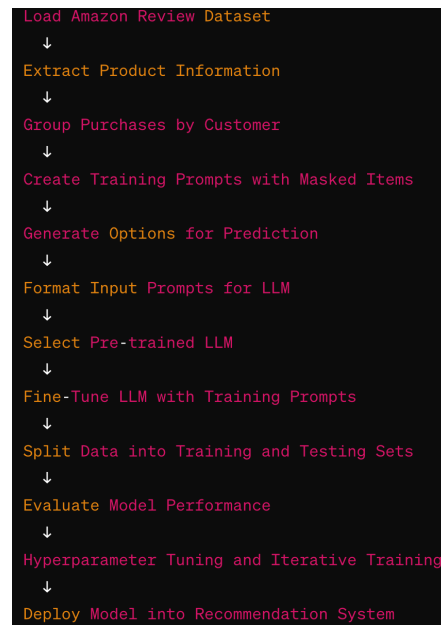


Figure 4: Flowchart for developing a text-based recommendation system by fine-tuning large language models (LLMs) using the Amazon Review Dataset

efficiency. Despite its smaller size compared to larger models, TinyLLAMA maintains high accuracy and speed, making it an excellent choice for applications requiring swift and effective language processing. We utilized TinyLLAMA to perform the same recommendation tasks, ensuring a balance between computational efficiency and predictive performance.

8 Implementation

For both models, classic deep learning training approaches were followed by attempting to overfit on a small subset of training data with no regularization (weight decay). Once overfitting was achieved, regularization was added until a high enough - but not too low - of a value was achieved while still converging on the loss. As mentioned in section 6.3, Mistral-7B was a robust model, but its computational demands were too great and impeded the rate of progress. In an effort to get around the hardware demands, pretrained Mistral-7B was used to produce inferences on all of the test set. As seen in Figure 5, the results were horrendous. Upon analysis of the actually outputs themselves, they were even worse. The model elected to respond in binary with long strings of 0s and 1s. Or it generated nonsensical outputs and appended "safe" words like "the", "or" and "and." This was interesting to see as it has been demon-

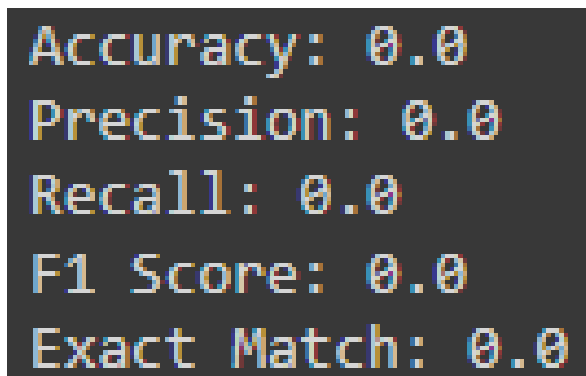


Figure 5: Pretrained Mistral Evaluation Metrics

strated large language models will output long outputs with "safe" words to maximize the probability of their output being considered "correct."

TinyLlama was hence trained and had great results. Small increments and decrements were done for tuning hyperparameters and a learning rate of .001 and weight decay of .001 were shown to be superior. We kept the batch size at an acceptable size of 8 to prevent potential VRAM overloads and we split the RAM workload among 4 processes. Considering dataset size and the narrower task scope, it was hypothesized and proven that a relatively smaller model would be able to handle the assignment and complete it at a highly accurate rate. This aligned with the project goal – create a model that can very accurately predict what a customer will elect to purchase next.

9 Results

The TinyLlama model displayed great results with very high rates of accuracy, precision, and recall. It trained much faster than Mistral-7B and clearly completed the task of choosing most likely (and correct) next item purchase by the user. This was incredibly interesting to see as many papers, media, and talks discuss the superiority of large language models. They promulgate that their superior memory, compute, and parameter capabilities make them better than smaller language models in nearly every way. The results of the training and testing of the TinyLlama model illustrate that this is not necessarily true. Bigger is not always better. This model had the compactness to learn the task it was given quickly and specialize in completing it.

Epoch	Training Loss	Validation Loss
1	0.627200	0.691961
2	0.365500	0.525095
3	0.374600	0.423947
4	0.320800	0.372932
5	0.283600	0.355217

Figure 6: Validation loss in evaluation metrics table

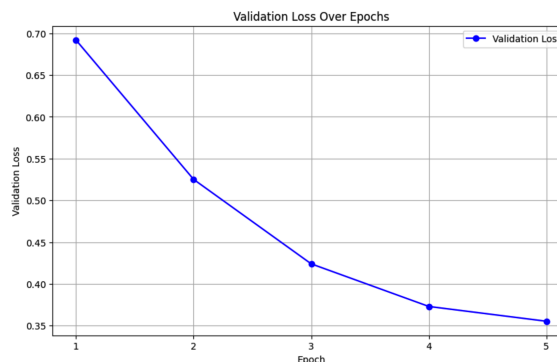


Figure 7: Validation loss in evaluation metrics graph

9.1 Evaluation Metrics

Many evaluation metrics were used to determine how successful the model was. As seen in Figure 8, the five evaluation metrics we showcased were Accuracy, Precision, Recall, F1 Score, and Exact Match. The high accuracy shows that the model was able to make the correct next item prediction for the given customer at an incredibly high rate of correctness. The high precision rate shows that the model is able to repeatedly predict similar results. The high recall rate shows that the model outputs highly relevant results. The high F1 score exemplifies the balanced performance of the model. Because the F1 score depends on the precision and recall, and because the model performed well for both of those metrics, the F1 score reflected both of these aspects. The high exact match score, which is a stricter version of accuracy, shows that the model was able to correctly predict the customer's next item with an overwhelming majority of the predicted items matching exactly with the ground truth labels. All of these metrics (each at over 98%) combined show exactly how robust the model was at precisely predicting which exact product a customer is likely to purchase next.

```
Accuracy: 0.9887051230334812
Precision: 0.9897135941912061
Recall: 0.9887051230334812
F1 Score: 0.9890547263681593
Exact Match: 0.9887051230334812
```

Figure 8: Evaluation Metrics

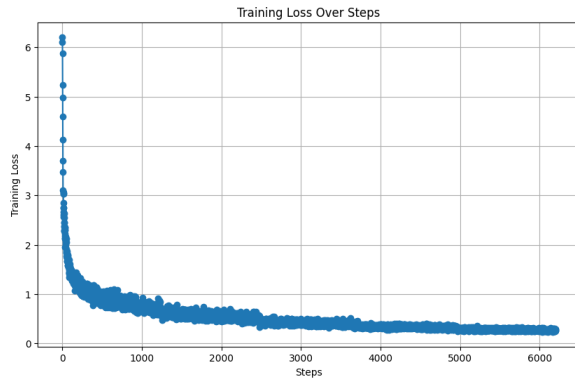


Figure 9: Loss Plot

9.2 Model Results

The model had great results, first of which is the successful convergence of its loss as seen in Figure 9. Over the over 6000 steps of training, the loss exponentially, then smoothly reduced to just barely above zero. This was an great result and setup the remaining capabilities of our project. The number of training points we covered were enough to demonstrate the major loss pattern but based on the plot, the loss would have continued to gradually decrease given more training data.

10 Error analysis

Figure 10 contains an example of an input the baseline failed at. In this example, the model ostensibly saw the nature-oriented quality of the past purchases by the user. The user bought products with keywords like White Premium Chicken Chunk, Ocean Spray, Brown Sugar Cinnamon, and Chicken of the Sea. All of these natural keywords could’ve caused the model to choose “Wood All Natural Root Beanut” as the customer’s likely next item because this item is also full of nature-oriented keywords. The correct recommendation in this example was actually “Original Ounce,” which the model likely oversaw due to the inundation of natural terminologies. There were three counts of the term “ounce” in the input prompt, but that was evidently overshadowed by nature-related

```
<? Below is an instruction that describes a user's interaction history with products on Amazon, write a response that appropriately completes the request.\n\n### Instruction:\nreviewer has bought Milk Premium Chunk Chicken Omeo Pack of, Ocean Spray Dried Omeo, Pop Breakfast Toasted Frosted Brown Sugar Cinnamon Family Count, Chicken of the Sea Premium Skinless and Boneless Pink Omeo, Considering their purchase history, which product will they buy next from the below options? A) Dark and Lovely All Natural Anti-Breakage Root to Tip Mender, 4 Fluid Ounce, B) Woodstock All-natural Almond Butter, Smooth, Unsalted, 16 oz, C) Pop-Tarts Toasted Pastries, Frosted Blue Raspberry, 8 Count, D) Original Omeo.\n\n### Response:\nMkood All Natural Root Beautu/s?"]
```

Figure 10: Example of Incorrect Recommendation Prediction

terms. This indicates additional attention heads and parameters could be of use to even further improve the accuracy of the model, though the 98% accuracy is already high enough. A larger model with more memory, compute, and attention could memorize and apply attention to the "ounce" keywords and potentially enable the model to predict the user's next item correctly. While errors were uncommon with our final model, in the incorrect cases, this was a common theme. We uncovered this by manually analyzing the error cases from our test set. Input prompts would contain overarching semantic tendencies, and the model would choose the product that most closely matched that. In these corner cases, the customer bought something that did not match the semantic tone of their past purchases, and thus the model was not able to recommend them the correct next item.

11 Contributions of group members

Each member of the group participated in all steps of the project. Together, we determined a project topic we were all interesting in working on. We then split the task into two main clusters: dataset acquisition/formation and model research/implementation. Two members of the group were assigned to the two clusters. This structure maintained for a while, until we reached a point in the project in which we began working together on all aspects of the project, and then began contributing to all aspects as well. Thus, there was not a clear separation of tasks. A general and loose surmise is bulleted below.

- Hemangani Nagarajan: data collection/processing, model training, paper writing
- Adhrit Srivastav: researched model options, built and trained models, paper writing
- Ishan Bhardwaj: researched model options, built and trained models, paper writing
- Kien To: data collection/processing, model training, paper writing

12 Conclusion

This project blended various learned NLP ideas and methodologies. A major takeaway we have from this project is that fine-tuning presents lots of benefits over training a new domain-specific model from scratch. We showed in this project that even with a dataset that was plausibly exponentially smaller than the training set for our pre-trained model, we were able to shift the focus of this model to a specific downstream task with high performance. This provides a strong argument to choosing pre-trained models over re-training: smaller datasets and the combination of general knowledge (from pre-training) and domain-specific knowledge (from fine-tuning) instead of only focusing on the latter. We also display the huge benefits provided by QLoRA, which smartly preserves the weights of the pre-trained model and layers the newly-learned parameters through low-rank matrix multiplication. Without this procedure, it would have been extremely difficult to fine-tune our models quickly and we might have lost the opportunity to display that *only* the fine-tuning was responsible for our results.

We also learned that prompt engineering strategies can be difficult to decide on and are extremely important. We had to make the assumption that our model would be able to understand natural language well (another reason why we focused on larger models) in order to change its order of operation when changes (even minute ones) were introduced in our prompts. Fortunately, we saw that for better or worse, our pre-trained models were effectively able to track these changes and performance and predictions became tailored to each prompting strategy. In-context learning proved to be an extremely effective tool and laid the foundation for our fine-tuning processes.

We went through some difficulties as well in this project; particularly in deciding relevant information for our prompts and cleaning irrelevant details, deciding on a pre-trained model, and efficiently preserving compute.

13 Future Works

For future considerations, we wish to expand on the capabilities of our current prompting strategies and translate from probability comparisons to generation (as done in our initial attempts). For the former case, we believe that pre-trained models with a larger context size with the same fine-tuning

strategy (choosing the correct product) can be effectively used in cases where a user has a large catalogue of choices and needs help in making a decision with regard to their product interaction history. Ideally, we would want the model to predict not only a single product but multiple potential options, while also finding a balance between the right amount of options and too many options. This is a case wherein models like Mistral-7B can be used; it would definitely require more data points and diversity across them, as well as more compute power. For shifting our models to text generation, we believe that the major determinant for success is the quality of the preprocessed dataset. While the training itself is important, preventing minuscule details and formatting to appear in training data would allow the model to focus more on keywords rather than irrelevant details, and provide more user-friendly predictions. While we do note that user input during inference would potentially contain fine-grained details like version history, copyright information, and so on, we believe one could either add a prompting layer to convert this to essential text (which could potentially deal with an important loss of information) or somehow shift the attention mechanism to majorly focus on less technical information. Finally, we believe that it would be useful to compare the efficiency of simply generating the option/prediction the model decides on rather than providing an entire prompt with product options. There would be major temporal benefits if the model's prompt size was restricted during inference, but the trade-off is the loss of the model's reasoning to make such a prediction. Such a study would definitely have varying outcomes for different enterprises.

14 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.

Yes, GPT-4o

If you answered yes to the above question, please complete the following as well:

- If you used a large language model to assist you, please paste **all** of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.

- “Generate related works for this topic: [paste topic]” - related works section
- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?
 - your response here

References

- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Edward J. Hu, Yelong Shen, P. W. Z. A.-Z. Y. L. S. W. L. W. W. C. (2021). Lora: Low-rank adaptation of large language models.
- Gao, Y., Sheng, T., Xiang, Y., Xiong, Y., Wang, H., and Zhang, J. (2023). Chat-rec: Towards interactive and explainable llms-augmented recommender system.
- Geng, S., Liu, S., Fu, Z., Ge, Y., and Zhang, Y. (2023). Recommendation as language processing (rlp): A unified pre-train, personalized prompt predict paradigm (p5).
- He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee.
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. (2023). Mistral 7b.
- Lin, J., Dai, X., Xi, Y., Liu, W., Chen, B., Zhang, H., Liu, Y., Wu, C., Li, X., Zhu, C., Guo, H., Yu, Y., Tang, R., and Zhang, W. (2024). How can recommender systems benefit from large language models: A survey.
- Sileo, D., Vossen, W., and Raymaekers, R. (2021). Zero-shot recommendation as language modeling.
- Tim Dettmers, Artidoro Pagnoni, A. H. L. Z. (2023). Qlora: Efficient finetuning of quantized llms.
- Wang, X., Wu, L., Hong, L., Liu, H., and Fu, Y. (2024). Llm-enhanced user-item interactions: Leveraging edge information for optimized recommendations.
- Xinyuan Wang, Liang Wu, L. H. H. L. Y. F. (2024). Llm-enhanced user-item interactions: Leveraging edge information for optimized recommendations.
- Xue, H.-J., Dai, X.-Y., Zhang, J., Huang, S., and Chen, J. (2017). Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Conference on World Wide Web*, pages 395–404. International World Wide Web Conferences Steering Committee.
- Yuanxing Liu, Wei-Nan Zhang, Y. C. Y. Z. H. B. F. F. H. C. Y. L. W. C. (2023). Conversational recommender system and large language model are made for each other in e-commerce pre-sales dialogue.
- Zhang, P., Zeng, G., Wang, T., and Lu, W. (2024). Tinyllama: An open-source small language model.