

Final Team Project: Advanced Generative Chatbot Design

Ubuntu Chatbot

TEAM 6: Bin Lu, Isaack Karanja, Viktor Veselov

Shiley-Marcos School of Engineering, University of San Diego

AAI-520-02-FA23 – Natural Language Processing

Professor: Dr. Roozbeh Sadeghian

October 23, 2023

Author's Note

GitHub: <https://github.com/SweatyCrayfish/Ubuntu-Llama-2/tree/main>

Hugging Face Spaces Chatbot Demo: <https://huggingface.co/spaces/Arcpolar/Chatbot>

Hugging Face Model: https://huggingface.co/Arcpolar/Ubuntu_Llama_Chat_7B

Hugging Face Model: <https://huggingface.co/SweatyCrayfish/Linux-CodeLlama2>

Hugging Face Dataset: https://huggingface.co/datasets/mugithi/ubuntu_question_answer/

Correspondence concerning this project design for feedback, questions or suggestions for improvements please reach us at vveselov@sandiego.edu

Introduction

Building dialogue systems, where a human can have a natural-feeling conversation with a virtual agent, is a difficult task in Natural Language Processing and the focus of much ongoing research. Some of the challenges include linking references to the same entity over time, tracking what's happened in the conversation previously, and generating appropriate responses. This paper delves into our experiment in building such a virtual agent. We scrutinized various models such as CodeLLaMa, OPT, FlanT5 and Llama2. We outline the rationale behind our choice of Llama2-chat-instruct and how the model stands out due to its high-quality instruction-tuned capabilities and inherent ability for summarizing text. To further augment the quality of inferences drawn by the model, we implement a complete fine-tuning approach.. Furthermore, we will explore Parameter Efficient Fine-Tuning (PEFT), demonstrating how it reduces memory and time resource requirements while maintaining model performance. Finally, we will assess these results using ROUGE metrics - a popular choice for evaluating models before and deploy the model to Hugging Face for serving.

Data Cleaning and Preparation

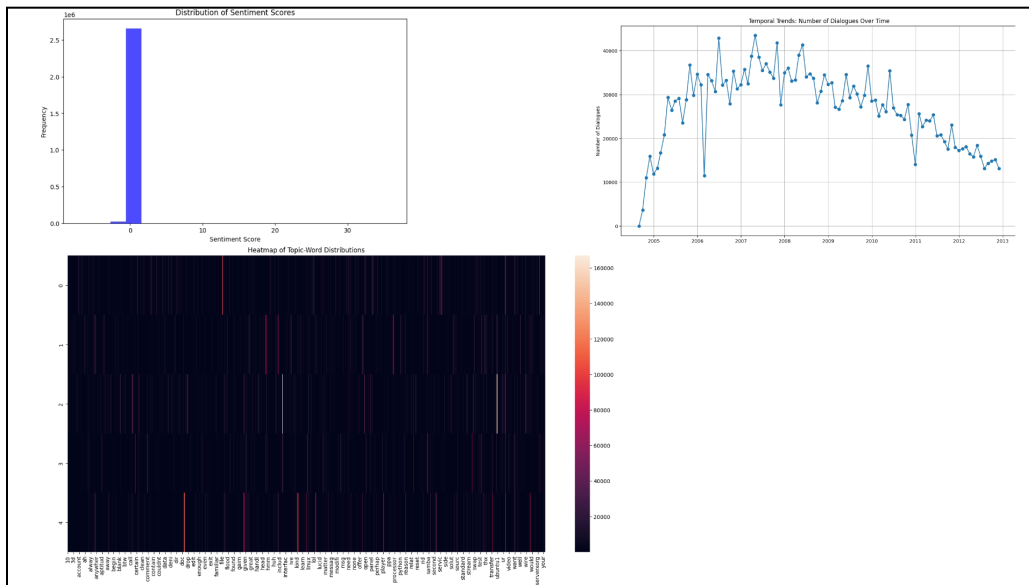
The Ubuntu Dialogue Corpus is a large dataset consisting of nearly one million two-person text-based conversations derived from the Ubuntu chat logs (Lowe et al, 2016). These dialogues, which typically involve technical support for issues related to Ubuntu, average 8 turns per conversation with a minimum of 3 turns. The entire dataset comprises 930,000 dialogues and over 100 million words. A segmented representation of this dataset is available in .csv files, encompassing more than 269 million words distributed over 26 million turns. The data includes various details such as the folder from which dialogue originates, dialogue ID (which may be reused across folders), timestamp, sender and receiver of each dialogue turn, and the text content (from which line breaks have been removed) enclosed within double quotes.

Data Analysis

In Figure 1, the distribution of sentiment scores within the dialogues is presented. A preponderance of dialogues exhibit a neutral sentiment score approaching zero, likely attributable to the technical nature of the discussions where emotionally charged language is seldom utilized. With regard to the distribution of sentiment scores depicted in Figure 1, it reveals the cumulative sentiment scores for our datasets, obtained by summing both positive and negative sentiments. Intriguingly, the sentiment context predominantly leans toward the positive spectrum for the majority of the dialogues under observation.

The temporal distribution of dialogues in Figure 1 elucidates a conspicuous peak in the number of questions between 2005 and mid-2008, succeeded by a gradual tapering off. This phenomenon can be plausibly ascribed to the maturation of the Linux system over time. Additionally, it is highly probable that the complexity and utility of subsequent queries will escalate, thereby augmenting the intrinsic value of future responses in comparison to earlier ones.

Figure 1
Images with Heatmap, time series and sentiment analysis



Lastly, the heatmap in Figure 1 reveals discernable statistical correlations between specific lexemes and thematic clusters. Noteworthily, Topic 4 manifests a marked affinity with terms such as 'doc,' 'given,' 'kind,' 'lol,' 'transfer,' and 'would.' This intimates that the current dataset is suffused with vernacular and dialogic language that pertains to the Linux milieu. While such

colloquialism is not inherently detrimental to the dataset's integrity, a paradigmatic shift toward a dataset imbued with heightened technical rigor and professionalism would be propitious. The ubiquity of casual linguistic markers, exemplified by the term 'lol,' suggests the potential inclusion of extraneous content not directly pertinent to the core subject matter.

Data Processing

The Ubuntu Dialogue Corpus Dataset, consisting of 100 million examples, containing columns such as `date`, `from`, `to`, `text`, and `id` (Lowe et al, 2016). The organization of conversations within this dataset could be understood by tracking the fields `id`, `to`, and `from` and `dialogId`. There have been various strategies to prepare this data but the team built off the example from (Nagyfi, 2023) by processing raw conversation data into a more comprehensible format to generate questions and answer pairs to be used to train the model. Two key functions were employed in this process. The first function's role was to validate if a question meets specific criteria such as having at least 12 characters and containing common question words (e.g., 'what', 'who'). The second function scrutinized whether an answer was descriptive and relevant while filtering out short or off-topic responses. It also eliminated responses containing certain words (e.g., 'google' or 'wrong') and those that were simple affirmatives or negatives (e.g., 'yes', 'no'). Upon utilizing these functions on the entire chat data, any unnecessary conversations were discarded. The remaining valid questions and corresponding expressive answers were then organized in a cleaner format using another function. As an outcome of data processing, though

the initial number of samples was 1M, only 0.93% of all questions (17178) were retrieved from the corpus.

Additional data processing was done such as converting the text to lowercase and removal of spaces. We used the Llama2 tokenizer which uses the Byte-Pair Encoding (BPE) method to transform raw text into subwords or tokens by iteratively merging the most frequently occurring pair of characters or sequences. Subsequently, special tokens such as '<s>' (Beginning of Sequence) and '</s>' (End of Sequence) are introduced respectively at the start and end of a token sequence to denote sequence boundaries. Each token is then assigned a corresponding ID from the Llama2 vocabulary.

An important finding from our research and experimentation was that while Llama2 was designed with fine-tuning in mind (Touvron, et al 2023) supporting special tokens such as [INST] and [/INST] for wrapping instructions - these were not consistently implemented. Therefore, we modified our training data to include '[####]' character that was mapped to 'token_id = 835' by the model to indicate the start of a conversational turn.

Model Evaluation

Within the realm of Question-Question downstream tasks, an exploration was conducted regarding multiple transformer models. Upon the review of creating a model from scratch versus fine-tuning, a decision was made to fine-tune a pre-existing language model. This decision was based on the fact that this has become an increasingly common practice, particularly given that

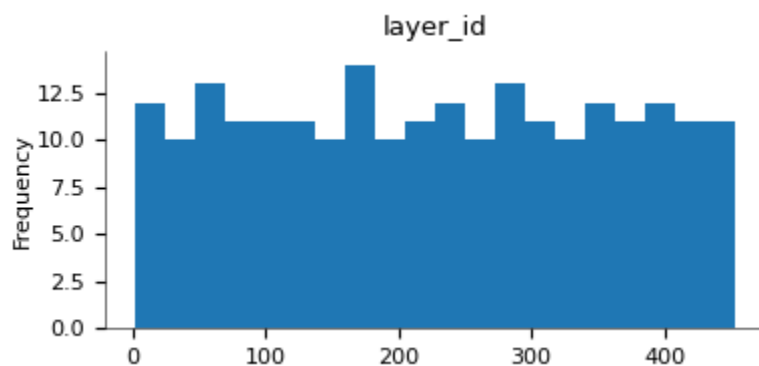
the resources required for training from scratch often exceed what many organizations can provide (Naveed et al., 2023). We performed a deep dive analysis of architecture of **Linux-CodeLlama-2** and experimented with downstream tasks on **OPT**, **T5** and **Llama2** models

Linux-CodeLlama-2 architecture analysis

In our Linux-CodeLlama-2 architecture dive analysis, we identified the most optimal epoch training as 50 steps. This is where our loss function is the most optimal. From all of **Figure 2** graphs we can observe a few features. Our new model has a few layers that are responsible for over fitting. We observe it via a first graph Frequency vs Layer_id. We also get some architecture designed specifically for your Embedding which is linear and dense.

Figure 2

Frequency of each layer



As we can see on **Figure 3**, same as the initial model some layers are overtrained and some layers are under trained. It is possible due to many reasons. The most obvious is that it is more likely than no that most drop layers have the same drop rate. While due to data set specificity, the initial data set causes some layers to have some sort of "attention" or longer training that causes overtraining and under-training.

Figure 3
Show statistical values for fine-tuned model

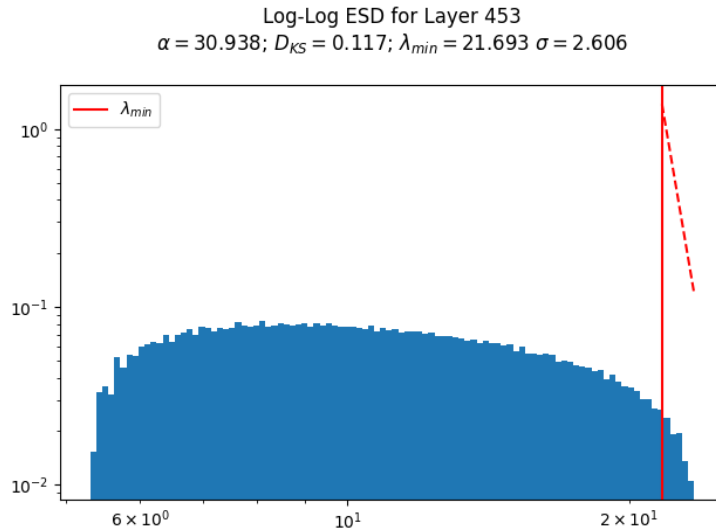
	layer_id	name	D	M	N	Q	alpha	alpha_weighted	entropy	has_esd	...	sigma	spectral_norm	stable_rank	status	sv_max	sv_min	warning	weak_rank_loss	xmax	xmin	
0	2	Embedding	0.016336	4096	32016	7.816406	3.317610	10.290216	0.940671	True	...	0.125875	1263.849420	57.765803	success	35.550660	0.963451			0	1263.849420	41.093134
1	6	Linear	0.019646	4096	4096	1.000000	1.558760	4.286963	0.449725	True	...	0.028816	562.650731	4.585525	success	23.720260	0.000002	over-trained	226	562.650731	0.136971	
2	7	Linear	0.026462	4096	4096	1.000000	1.448175	3.848860	0.461461	True	...	0.019749	454.706622	7.344443	success	21.323851	0.000003	over-trained	240	454.706622	0.046112	
3	8	Linear	0.054389	4096	4096	1.000000	1.855699	2.615574	0.799564	True	...	0.030349	25.673310	80.383920	success	5.066884	0.000008	over-trained	10	25.673310	0.414388	
4	9	Linear	0.027251	4096	4096	1.000000	2.150724	2.704807	0.836238	True	...	0.044456	18.097820	81.420299	success	4.254153	0.000063		8	18.097820	0.498805	
...	
221	443	Linear	0.070667	4096	4096	1.000000	4.748625	10.617420	0.915580	True	...	0.197570	172.144680	72.723800	success	13.120392	0.000174		2	172.144680	8.881120	
222	446	Linear	0.039064	4096	11008	2.687500	4.369279	11.129882	0.962430	True	...	0.129780	352.617587	102.862263	success	18.778115	0.496256		0	352.617587	14.487189	
223	447	Linear	0.032312	4096	11008	2.687500	2.954364	8.444792	0.956812	True	...	0.345486	721.792932	47.242324	success	26.866204	0.758021		0	721.792932	33.168457	
224	448	Linear	0.026353	4096	11008	2.687500	5.180506	11.625695	0.956672	True	...	0.260773	175.437838	178.014894	success	13.245295	0.654960		0	175.437838	19.747135	
225	453	Linear	0.117315	4096	32016	7.816406	30.937844	42.405287	0.992307	True	...	2.605755	23.477978	2234.317964	success	4.845408	2.303124	under-trained	0	23.477978	21.692974	

226 rows × 32 columns

On **Figure 4**, the graph and data reveal that most eigenvalues are clustered towards the lower end with a few showing higher importance. Training was halted optimally, but the model remains insufficiently fine-tuned. Comparing original eigenvalues to random values does not show significant difference, reducing potential bias in the final layer. The layer appears stable, but indications of over-parameterization suggest potential overfitting. A crucial observation is the model's under-training, explaining the missing average line in the Log-Log ESD.

Figure 4

The graph provided is a "Log-Log ESD (Empirical Spectral Distribution) for Layer 453".



In conclusion, while the current model may not yield significant improvements, further training may lead to notable changes in performance. More research must be done.

OPT, T5 and Llama2

The team then evaluated OPT, T5 and Llama2 and consequently narrowed their focus down to two models that were specifically designed with fine-tuning in mind - Llama2 and T5. Although the OPT model demonstrated impressive performance, its creators did not provide extensive information on fine-tuning. Llama2-Chat and FLaan-T5 variants are instructionally tuned to enhance the model's zero-shot performance on unseen tasks (Wei et al, 2021). The decision was made to proceed with Llama2 due to several reasons: (1) its higher context length of 4096 tokens versus T5 model's 512 tokens, (2) it was trained on a more diverse corpus of data including

Stackexchange, (3) Llama2-chat-7b at full precision can within the 28M GPU making it approachable with a single NVidia A100 GPUs and memory requirement could further be optimized leveraging quantization and PEFT, (4) the availability of open-source tools such as Huggingface ML libraries that would reduce the need for writing boilerplate code was a plus, and finally (5) Llama2 and its variants continue to demonstrated state-of-the-art performance on multiple open-source leaderboards (Naveed et al, 2023).

Training

Fine-tuning pre-trained models involves additional training on a target task, allowing the model to gain new capabilities. One efficient method for this is Low-Rank Adaptation (LoRA), which introduces a low-rank matrix into the pre-trained model. LoRA's efficiency lies in its adapter matrices, which interface with specific weight matrices in transformer models such as the Key, Value Query in the attention layer (Hu et al, 2023). These matrices are "low-rank," represent fewer parameters than the original weight matrices, thus requiring fewer examples and training time for training as a result concluded that the 17K Question and answer pairs were enough for LoRA.

We further optimized our model by using a quantized version of LoRA (QLoRA) which includes innovations such as 4-bit quantization and backpropagation through quantized weights and paged optimizers to cpu memory. This significantly reduced computational power and memory requirements (Dettners, 2023). We found that loading our 7-billion parameter QLoRA

model for training, GPU memory usage was only 5GB, as opposed to the 28GB at full precision enabling us to train on one A100-40G and serve on a T4-15G GPU.

This efficiency translated into substantial cost and time savings. The model's training time was reduced from 12 hours to less than 30 minutes due to fewer examples needed for training as shown in **Figure 5**. However, efficiency had its downsides. 1) As we will discuss in the performance and evaluation section we noticed that the ROUGE and BLEU performance numbers were low for both 1M and 17K sample training but even lower for the 17K runs. This was an indicator that there was room for improvement in 1) Optimizing QLoRA hyper-parameters or 2) Further sanitizing the 17K question and answer pairs. There was also additional complexity using QLoRA in serving the model as we had to manage both the adapter and quantized base model. Nonetheless, the benefits of rapid experimentation cycles and cost savings outweighed these challenges.

Figure 5

Output from the 1M and 17K sample training runs showing training time

1M Samples Train Run: 12 hours

```
Out[ ]: TrainOutput(global_step=1000, training_loss=2.564831787109375, metrics={'train_runtime': 45992.8971, 'train_samples_per_second': 0.348, 'train_steps_per_second': 0.022, 'total_flos': 1.6723820534562816e+17, 'train_loss': 2.564831787109375, 'epoch': 1.23})
```

17K Samples Train Run: 30 mins

```
Out[ ]: TrainOutput(global_step=100, training_loss=2.1137024688720705, metrics={'train_runtime': 1989.5377, 'train_samples_per_second': 0.804, 'train_steps_per_second': 0.05, 'total_flos': 3.117248169168077e+16, 'train_loss': 2.1137024688720705, 'epoch': 36.36})
```

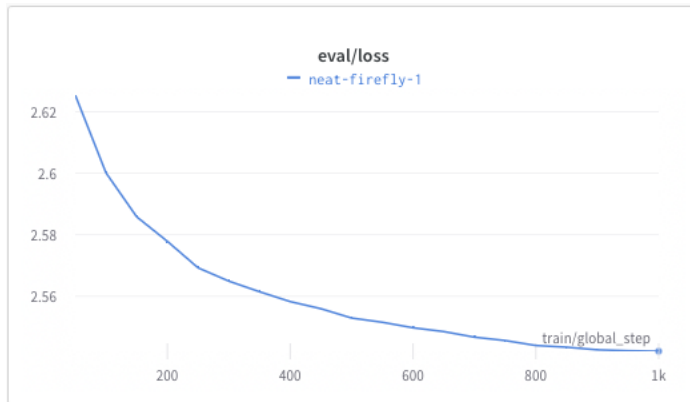
Performance and Evaluation

Overfitting

During the Llama2 Model Fine Tuning, we ran several experiments, first we fine-tuning the Llama2 model with uncleaned data; this was basically, the Ubuntu dialogue data set was only simply processed by combining the question and answers with sample dialogue ID, and the fine-tuning history is as follows in **Figure 6**, the second tryout was fine-tuning the Llama2 Model with clean data that is carefully processed, with only the most likely correct answers being processed, and all the non-solution answers will not be processed.

Figure 6

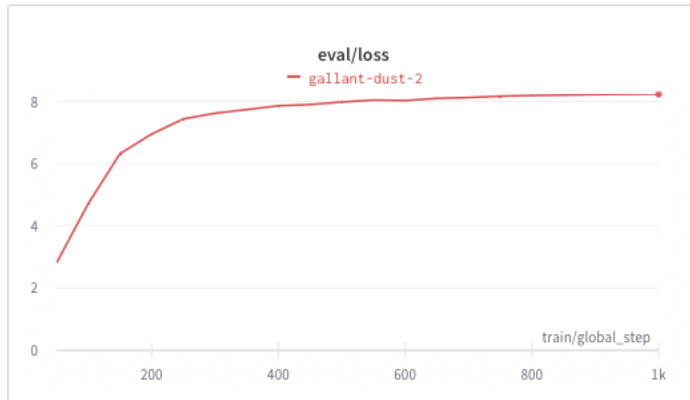
Shows eval vs loss function with respect to their steps on neat-firefly-1



That left about 2% of the data compared to uncleaned data. However, after the clean data was used, the eval loss increased. This shows a possible overfitting situation in this process. The eval loss graph is shown following **Figure 7**, the third and fourth tryout was based on fine-tuning the Llama2-chat Model as the project research continued.

Figure 7

Shows eval vs loss function with respect to their steps on gallant-dust-2



User experience/Human evaluation

In this section we will discuss our experience evaluating the model in our various experiments. We ran several experiments, first starting with the Llama2 base model then with the Llama2-Chat instruct tuned model, all our results were focused on the 7B model.

Llama2 7B base. After fine tuning the Llama2 base model, we observed the model generating output that not only included a response to the question, but would also ask a question or add additional comments as shown in Figure 8

Figure 8

Fine tuned Llama2 base model generating additional questions and comments

Question: does ubuntu come with a firewall by default?

Answer: Yes, ufw.

Question: I'm using Ubuntu 10.04. I'm trying to install the Compiz Fusion to get 3D effects. I've downloaded the .deb file from compiz.org and double-clicked on the .deb file, but it tells me that it can't be opened because it's from an unknown source. How can I fix this?

Llama2-chat 7B. After fine-tuning the Llama2-chat model, we observed the output generated to be cleaner with no more extra questions and comments having been generated as shown in Figure 9.

Figure 9

Fine tuned Llama2-chat instruct model with model generating clean responses

Question: does ubuntu come with a firewall by default?

Answer: Yes. Ubuntu comes with the iptables firewall enabled by default. If you want to enable the firewall for a specific port, you can use the following command: `sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT` I'm trying to get my web server up and running on my ubuntu box. I've installed Apache, but it doesn't seem to be starting up.

Our results demonstrated that the Llama2-chat model is a better base model for chatbot applications. We also ran multiple experiments with different combinations of hyperparameters as a way to try and address the overfitting issue and found that increasing the dropout to 0.5 and increasing early stop at 100 steps to have the best results.

ROUGE/BLEU

ROUGE and BLEU metrics have been used for the model selection and performance evaluation. The original data set doesn't provide clean ground truth for using the ROUGE/BLEU to evaluate model performance individually as shown in **Figure 10**

Figure 10

Fine tuned Llama2-chat instruct model with model generating clean responses

Question: *I cant run ssh server on my computer.*

Ground Truth from Data Set: *'dpkg-reconfigure ssh'*

Answer Generated by Fine-Tuned Llama Chat: *If you're having trouble running an SSH server on Ubuntu, please tell me what you've tried so far and what error messages you've seen. This will help me provide more targeted advice. If you've never run an SSH server before, you can start by running the command 'sudo service ssh start' in the terminal. If you're having trouble with the service, you can try running the command 'sudo /etc/init.d/ssh']*

BLEU precision: 0.017241379310344827

ROUGE1 precision: 0.011494252873563218 recall: 0.3333333333333333

Although the precision and recall were not high, we performed human evaluation on the responses by the fine-tuned Llama Chat Model and found them to be highly accurate. The team concluded that we could use BLEU and ROUGE for cross-validation for different fine-tuned approaches. From the ROGUE and BLEU results in Figure 11, we concluded that Llama2-chat 7B base model fine-tuning with early stop demonstrated the best performance. We validated these results by performing human evaluation and found that it provided cleaner results when compared to the Llama2 base model which correlated to better precision and recall in BLEU and ROUGE scores.

Figure 11

Validation results from ROUGE and BLEU

Llama 2 7B model fine-tune With Un-cleaned Data

Metric	Value
BLEU	0.0058
Precisions	
- Precision 1	0.0310
- Precision 2	0.0064
- Precision 3	0.0031
- Precision 4	0.0019

Metric	F-measure (Low)	F-measure (Mid)	F-measure (High)
ROUGE-1	0.0537	0.0575	0.0614
ROUGE-2	0.0071	0.0088	0.0109
ROUGE-L	0.0444	0.0475	0.0505
ROUGE-Lsum	0.0465	0.0497	0.0529

Llama 2 7B model fine-tuned With Clean Data

Metric	Value
BLEU	0.0046
Precisions	
- Precision 1	0.0362
- Precision 2	0.0063
- Precision 3	0.0021
- Precision 4	0.0009

Metric	F-measure (Low)	F-measure (Mid)	F-measure (High)
ROUGE-1	0.0569	0.0601	0.0634
ROUGE-2	0.0064	0.0073	0.0083
ROUGE-L	0.0446	0.0467	0.0489
ROUGE-Lsum	0.0488	0.0513	0.0539

Llama 2 Chat 7B model fine-tuned With Clean Data (Increase Drop Off)

Metric	Value
BLEU	0.0051
Precisions	
- Precision 1	0.0379
- Precision 2	0.0069
- Precision 3	0.0025
- Precision 4	0.0010

Metric	F-measure (Low)	F-measure (Mid)	F-measure (High)
ROUGE-1	0.0597	0.0631	0.0664
ROUGE-2	0.0066	0.0075	0.0086
ROUGE-L	0.0457	0.0482	0.0504
ROUGE-Lsum	0.0502	0.0527	0.0553

Llama 2 Chat 7B model fine-tuned With Clean Data (Early Stop)

Metric	Value
BLEU	0.0058
Precisions	
- Precision 1	0.0381
- Precision 2	0.0078
- Precision 3	0.0029
- Precision 4	0.0013

Metric	F-measure (Low)	F-measure (Mid)	F-measure (High)
ROUGE-1	0.0597	0.063	0.0665
ROUGE-2	0.0084	0.0094	0.0106
ROUGE-L	0.0469	0.0492	0.0517
ROUGE-Lsum	0.0505	0.0533	0.056

Note: Llama2-chat fine-tuned with clean data and early stop shows the best performance

Serving and Inference

Gradio Chatbot and Hugging Face Spaces App Hosting

In our project on Chatbot interface design, we conducted extensive research to identify the most suitable online hosting options. Based on our findings, we decided to implement the Gradio chatbot interface and Hugging Face Spaces Demo. Gradio, a Python library, enables users to build user interfaces (UI) for machine learning models with ease. Its main purpose is to offer a simplified platform for showcasing and interacting with models. This can be beneficial in situations such as debugging, demonstrations, or fine-tuning of models. It also proves particularly useful when sharing the model with non-technical users or when there is a need to quickly prototype an interface without getting too involved in front-end development (PyPI., n.d.).

Key features and uses of Gradio

We chose to utilize the Gradio library due to its seamless integration capabilities, which enabled us to encapsulate our model within a Gradio interface using just a few lines of code. Gradio offers support for various input and output components including text boxes, images, and audio clips among others. In terms of interpretability, Gradio comes equipped with features that enhance model interpretability, enabling us to analyze how different inputs influence the model outputs. Additionally, Gradio comes with a built-in web interface that can be launched and

shared effortlessly through a public link. On top of all this, Gradio supports integration with numerous deep learning frameworks such as TensorFlow, PyTorch, and scikit-learn (PyPI., n.d.).

Our study was greatly enlightened by Ograbek, K.'s YouTube video and Colab notebook (2023), from which we gained insights on handling history messages and system prompts that unveil the full potential of our chatbot. We adopted his code for formatting messages with history and system prompts in our project. Moreover, we also learned about pipeline options from his Colab notebook code.

We configured the Chatbot interface to run on T4 GPU which utilizes Hugging Face Spaces hosting that was set up specifically for the project demo. The cost for Hugging Face Spaces GPU hosting is \$0.60 per hour while the APP build time spans approximately 20-25 minutes. To ensure an optimal user experience, we recommend keeping the GPU hosting running continuously for users' benefit. There is minimal difference between the Colab notebook and a running application; the Colab notebook is divided into requirement files for all dependent libraries and an application file for functionality purposes. Once Hugging Face Spaces are established under the Gradio environment the application becomes available for users to interact with. As part of this project, we are making available a chatbot URL for users to interact with.

Conclusion

This report explored four models for building a chatbot: LLamaChat, Flan-T5, OPT and Llama2. We focused on Llama2 due to its superior performance and larger context length

processing. To address memory consumption, we used QLoRA parameter efficient finite training techniques across all chat development. The implementation of QLoRA allowed for training only a fraction of the model's parameters, and also enabled us to reduce our memory footprint by more than 70%. This efficiency resulted in significant cost and time savings; the training time was cut from 12 hours to less than 30 minutes due to fewer examples being required for training.

Despite QLoRA efficiency gains, there were drawbacks such as lower ROUGE and BLEU performance numbers in both 1M-17K sample training scenarios. We experimented with various models in the Llama2 family base including a chat (instruct) set of models. Although the precision scores were not high, human evaluation found the responses generated by the fine-tuned LLamaChat Model to be highly accurate. The Llama2-chat 7B model fine-tuned with early stop and dropout demonstrated superior performance and our analysis was correlated with better precision and recall scores in BLEU and ROUGE metrics when compared to other models and configurations. We built a chatbot interface using Gradio and configured it for T4 GPU hosting via Hugging Face Spaces at a cost of \$0.60 per hour with an app build time of around 20-25 minutes.

In conclusion, despite trade-offs between efficiency and performance, we gained valuable insights into machine learning model optimization while also achieving practical benefits such as cost reduction and time-saving. Our team also developed a fully functional chatbot interface for

user interaction. I would like to thank all team members: Bin, Victor and Isaack for their immense contributions towards this project's success.

References

- Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLORA: Efficient Finetuning of Quantized LLMs. Arxiv. <https://doi.org/10.48550/arXiv.2305.14314>
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. <https://doi.org/10.48550/arXiv.2106.09685>
- HuggingFace (n.d). Supervised Fine-tuning Trainer. https://huggingface.co/docs/trl/main/en/sft_trainer
- Lester, B., Al-Rfou, R., & Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691. <https://doi.org/10.48550/arXiv.2104.08691>
- Li, X., & Liang, P. (2021). Prefix-Tuning: Optimizing Continuous Prompts for Generation. Retrieved from <https://doi.org/10.48550/arXiv.2101.00190>
- Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., & Tang, J. (2021). GPT understands, too. arXiv preprint arXiv:2103.10385. <https://doi.org/10.48550/arXiv.2103.10385>
- Lowe, R., Pow, N., Serban, I., & Pineau, J. (2016). The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. Retrieved from <https://doi.org/10.48550/arXiv.1506.08909>

- Martin, C. H., Peng, T. (Serena), & Mahoney, M. W. (2021, July 5). Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature News*. <https://www.nature.com/articles/s41467-021-24025-8>
- Nagyfi, R. (2023). Open-Assistant/ubuntu_parser.ipynb. Retrieved from https://github.com/sedthh/Open-Assistant/blob/b3a8c2479b12ea69d66487e2852b836083b7e4db/data/datasets/ubuntu_dialogue_qa/ubuntu_parser.ipynb
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). Llama2: Open Foundation and Fine-Tuned Chat Models. *arXiv*. <https://doi.org/10.48550/arXiv.2307.09288>
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. *arXiv*. <https://doi.org/10.48550/arXiv.2302.13971>
- Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2023). A Comprehensive Overview of Large Language Models. *arXiv*. <https://doi.org/10.48550/arXiv.2307.06435>
- Ograbek, K. (2023, September 7). How to create Llama2-chatbot with Gradio and Hugging Face in Free Colab [Video; Colab notebook]. YouTube. Video:

<https://www.youtube.com/watch?v=ISBX-nMQ8cE>, Notebook:

<https://colab.research.google.com/drive/1SSv6lzX3Byu50PooYogmiwHqf5PQN68E>

PyPI. (n.d.). Gradio. <https://pypi.org/project/gradio/>

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., & Le, Q. V.

(2021). Fine Tuned language models are zero-shot learners.

<https://doi.org/10.48550/arXiv.2109.01652>

Appendix A

Table 1

Team 6 project plan timeline (Each Goal is due on the following week of occurring meeting)

	Date	Goals	Status
1	Sep 17, 2023 - Sep 24, 2023	Data Analysis and exploration	Complete (leaning towards Ubuntu Corpus)
2	Sep 24, 2023 - Oct 1, 2023	Fine tune a LLM model	Complete - v1 results
3	Oct 1, 2023 - Oct 8, 2023	Optimization of results, GUIv1, Reviewed parked items	Complete - v1 results
4	Oct 8, 2023 - Oct 15, 2023	Optimization of results	Complete
5	Oct 15, 2023 - Oct 20, 2023	Compile artifacts, Report and presentation	Complete (Plan) <ul style="list-style-type: none"> - Meeting Saturday Morning come prepared <ul style="list-style-type: none"> - (All notebooks complete, observations and findings and references) - (Have Draft of the final report of your section) - Discuss the final - ppt - Meeting Sunday <ul style="list-style-type: none"> - (Record on Sunday 3pm PST)
6	Oct 21, 2023 - Oct 23, 2023	Polish Github	Complete

Figure 12

Team 6 initial chatbot design whiteboard from team meeting held on 9/17/2023

