Personalised Chat Bot (CHADAI) Development Report

Introduction

In the realm of artificial intelligence (AI), natural language processing (NLP) has witnessed remarkable advancements, enabling machines to understand and generate human-like text. Chatbots, in particular, have become increasingly prevalent in various domains, serving as virtual assistants, customer support agents, and companions in daily interactions. However, traditional chatbots often lack the depth and personalization needed to truly engage users in meaningful conversations.

This project endeavours to bridge this gap by leveraging state-of-the-art Machine Learning eXchange (MLX) models and WhatsApp chat data to create a highly personalised chatbot experience. By fine-tuning the Mistral 7B model, renowned for its proficiency in language understanding, on individual users' WhatsApp chat histories, we aim to develop a chatbot, affectionately named CHADAI, capable of providing responses tailored to each user's unique conversational style and preferences.

Literature Review

The evolution of chatbot development traces back to the 1960s, with ELIZA, a primitive AI designed to simulate human conversation. Over the decades, advancements in natural language processing and machine learning fueled the growth of chatbots. Today, they serve diverse purposes, from customer service and virtual assistants to healthcare and education. Chatbots streamline communication, enhance user experiences, and offer round-the-clock support. Their integration into messaging platforms, websites, and mobile apps has revolutionised interactions, making them an indispensable tool in the digital age, bridging the gap between humans and technology with unprecedented efficiency and convenience.

The journey of chatbot development intertwines with the rise of personalised AI models, where early endeavours like ELIZA laid the groundwork in the 1960s. Over time, advances in machine learning and natural language processing refined chatbots, leading to sophisticated personalised AI models. Today, they revolutionise various sectors, from e-commerce to healthcare, offering tailored experiences and efficient problem-solving. With capabilities to understand context, preferences, and emotions, personalised AI chatbots redefine customer interactions, streamline processes, and augment human productivity. Their ubiquitous presence in messaging apps, websites, and virtual assistants underscores their pivotal role in modern communication and user engagement.

Fine-tuning techniques play a crucial role in enhancing personalised AI chatbots. Leveraging large datasets, fine-tuning optimises model parameters to adapt to specific tasks or domains. Techniques like transfer learning enable chatbots to learn from existing knowledge, while continual reinforcement ensures ongoing refinement. These methods empower chatbots to deliver highly customised and contextually relevant interactions.

Motivation:

Current chatbots are handy but often fail to engage in meaningful chats, leaving users with generic answers that don't reflect their communication style. By integrating MLX models with WhatsApp chat data, we aim to transform the chatbot experience into one that deeply understands each user's language patterns, preferences, and personality traits. This approach is expected to create a stronger connection and rapport between the user and the chatbot.

Additionally, this project addresses the need for AI technologies to mirror the diverse ways humans communicate. In a digital-dominated world, it's increasingly important to interact with AI as naturally and intuitively as with a human. CHADAI will explore how MLX models can shift from rigid, scripted responses to dynamic, meaningful interactions, leveraging WhatsApp data to push the boundaries of conversational AI and unlock new avenues for personalised human-machine dialogue.

Methodology

1. **Data Collection:**

   Users are instructed to export their WhatsApp chat histories from the application settings. These chat logs serve as the training data for fine-tuning the MLX model.

   To create our personalized chatbot, we needed chat data in English, but we encountered a little hiccup. You see, people tend to mix languages when they chat, so finding proper English data was like looking for a needle in a haystack. What did we do? Well, we formed a little trio and chatted away to collect our own data! It was like a mini-chatting marathon. But wait, there's more! We also discovered a fun trick: we fine-tuned our chatbot on English movie scripts. Can you imagine? Now, our chatbot can dish out responses as if it's your favorite movie character chatting with you in real life! It's like having a movie star as your virtual buddy. Who knew building a chatbot could be this entertaining?

2. **Data Preprocessing:**

   We made a python script to make cleaner data, so that it gets fine-tuned properly.

Here's a detailed breakdown of each step of the algorithm:

1. Importing Necessary Libraries:
   - json for handling JSON operations
   - re for regular expressions (to manipulate strings based on patterns)
   - argparse for parsing command-line arguments

   These libraries are crucial for data handling and processing based on user inputs.

2. Defining the clean_and_format Function:

   This function takes three arguments namely, input_file, output_file, and max_split_len. These parameters define the source of the data, the destination for the processed data, and the maximum length of text chunks, respectively.

3. Identifying New Conversations:

   An inner function is_start_of_conversation uses a regular expression to detect timestamps typical at the start of a new chat message. This detection is pivotal for segregating individual messages or conversation starts.

4. Removing Timestamps and Handling New Lines:

   Another inner function remove_timestamp_and_convert_newlines strips the timestamp from each line and formats the text for consistent newline handling.

5. Reading and Chunking:

   The script reads from the input file line by line, applying the timestamp removal function. It then organizes the cleaned lines into chunks. A chunk represents a continuous text block that is cut off either at the start of a new conversation or when it reaches the specified maximum length (max_split_len).

6. JSONL Output:

   Each chunk of cleaned and formatted text is converted into a JSON object with a key "text" and written into the output file in JSONL format (JSON Lines), where each line is a valid JSON object.

7. Defining the split_jsonl Function:

   It takes parameters for the input JSONL file, files for test, validation, and training splits, and sizes for these splits.

8. Data Splitting Logic:

After reading all lines from the input JSONL file, the function checks if there are enough entries to meet the specified sizes for test and validation splits. It then allocates the beginning lines to test data, the following ones to validation data, and the remainder to training data.

9. Writing Split Data:

   The separated data chunks are written to their respective files (test, validation, and training).

10. Main Execution Block:

    Using argparse, the script parses command-line arguments which specify file paths and parameters like max_split_len. These parameters control how the functions are called to process and split the data.

(So, the Python script automates the preprocessing of chat data for analytical or machine learning applications. It cleans data by removing metadata like timestamps, formats it into a uniform structure, and splits it into manageable chunks. The core functionalities are wrapped in two main functions. clean_and_format reads chat data, processes it into a cleaner format, and chunks it based on conversation breaks or length limits, outputting JSONL files. split_jsonl takes this processed data and divides it into sets for training, validation, and testing, facilitating model development or further analysis. The script's use of command-line arguments enhances its adaptability, making it suitable for various data sizes and types. Overall, it's a robust tool for preparing raw chat data for deeper analysis or model training.)

{Keep the above para if page num is less.}

The script is designed for flexibility with defaults and optional arguments, allowing users to customize the processing based on their specific dataset size and format requirements.

3. Model Selection and Fine-Tuning:

When selecting a model for fine-tuning, factors like architecture, pre-training objectives, and task compatibility are crucial. Our project chose Hugging Face's Mistral-7B model for its robust architecture and versatility, ideal for WhatsApp chatbot development. Additionally, its quantization capability ensures efficient deployment on resource-constrained devices.

Mistral 7B v0.1, with 7-billion parameters, outperforms Llama 2 13B and Llama 1 34B in various tasks. Leveraging grouped-query attention for faster inference and sliding window attention for longer sequences, Mistral 7B excels in capturing nuanced language patterns. Its compatibility with MLX framework on Apple Silicon further enhances performance and efficiency. We thus use MLX to fine tune the model with LoRa.

4. **Interface**:

   Initially, our project aimed to integrate the chatbot interface directly with WhatsApp. However, we encountered significant complexities due to WhatsApp's stringent privacy measures and the challenges associated with modifying a closed-source application.

   So to demonstrate our model's capabilities, we have three types of interfaces built by us with which the model is integrated.

   One of them is a basic terminal interface where you can get inferences from the model using a single terminal command with a prompt.

   Second one is a GUI application which takes user input as prompts upon entering every character and generates and displays a response for the user to use in the input in his/her text with a single key press. Here the responses by AI are like personalised suggestions which was our basic idea.

   We also created a multi user chatting interface and integrated the model with it to sort of replicate a Whatsapp group environment. Here multiple users can chat in a chatroom, where our AI provides suggestions based on the user and other users' previous texts and what the user is currently typing. These suggestions can be used by the user by a single key press.

   The instructions to use the interfaces can be found in readme.txt .

Mathematical Model:

Mistral 7B leverages grouped-query attention (GQA) , and sliding window attention (SWA). GQA significantly accelerates the inference speed, and also reduces the memory requirement during decoding, allowing for higher batch sizes hence higher throughput, a crucial factor for real-time applications.
In addition, SWA is designed to handle longer sequences more effectively at a reduced computational cost, thereby alleviating a common limitation in LLMs. These attention mechanisms collectively contribute to the enhanced performance and efficiency of Mistral 7B. These models have complex execution mechanism. The actual working can be explored through this research paper provided in the references section.

Performance Metrics:

Train Loss:

- The training loss represents how well the model is fitting to the training data. A consistent decrease in training loss, as seen in the results, is indicative of the model's improved learning and adaptation to the nuances of the training data over each iteration. This trend confirms that the model is effectively minimising the error in its predictions during the fine-tuning process.

Validation Loss:

- Validation loss measures how well the model generalises to data it hasn't seen before. The observed minor fluctuations in validation loss suggest that the model is not overfitting, meaning it retains a level of generalization. However, these fluctuations also highlight potential opportunities for hyperparameter optimization to achieve even lower validation loss, thus enhancing the model's predictive performance on unseen data.

Iteration Speed (Tokens/second):

- The tokens processed per second provide insight into the computational efficiency of the model during training. The maintained high rate of tokens processed per second implies that the model is not only learning effectively but is doing so with computational efficiency. This balance is crucial when deploying models to production environments, especially when operating within resource constraints.

Comparative Advantage:

Model Comparison:

- When benchmarked against larger models such as Llama 2 13B and Llama 1 34B, Mistral-7B v0.1 shows superior performance in various tasks. This implies that despite having fewer parameters, the model does not compromise on the quality of interaction and can efficiently manage the complexity of language patterns found in WhatsApp chats. This advantage is particularly significant considering the balance between resource consumption and performance is a key factor in deploying AI models in production.

Language Adaptability:

- The model's ability to capture nuanced language patterns is crucial for a chatbot aimed at providing a personalized experience. Mistral-7B's design, incorporating grouped-query attention and sliding window attention mechanisms, contributes to its adaptability in processing and generating conversational language that feels natural and personalized to users.

Discussions:

Challenges:

1. **Limited Data Diversity:**
   WhatsApp chat data, while rich in conversational content, may exhibit biases or limited diversity in linguistic expressions, topics, and communication styles. This lack of diversity can hinder the model's ability to generalize beyond the training data, leading to suboptimal performance when confronted with novel conversational contexts.

2. **Overfitting Risk:**
   One of the primary challenges is the risk of overfitting, wherein the model becomes overly specialized to the training data and fails to generalize well to unseen data. Given the variability and complexity of human language, overfitting can occur if the model memorizes specific patterns in the training data rather than learning generalizable representations.

3. **Resource Constraints:**
   Training MLX models, especially on resource-constrained devices such as the MacBook Pro 14" with M1 Pro chip, poses additional challenges like it took about 5.5 hours for complete training (all 600 iterations) for me . Limited computational resources may restrict the model's capacity for extensive exploration of the parameter space and comprehensive learning of nuanced language patterns, potentially compromising its ability to generalise effectively.

Future Directions:

1. **User Feedback Integration:** Incorporating user feedback mechanisms allows the chatbot to continuously adapt and improve its responses over time based on user interactions.
2. **Multimodal Integration:** Integrating multimedia content from WhatsApp, such as images and voice messages, into the chatbot's understanding expands its capabilities beyond text-based interactions.
3. **Domain-Specific Adaptation:** Extending the fine-tuning process to incorporate domain-specific knowledge enhances the chatbot's performance in specialized domains, such as customer service or technical support.

Conclusion:

Integrating MLX models with WhatsApp chat data presents a promising approach to developing personalized chatbot experiences. By fine-tuning the Mistral 7B model on individual users' chat histories, CHADAI demonstrates the potential for MLX to create highly customized and engaging conversational agents.

Through careful data preprocessing, model conversion, and training, CHADAI learns to understand and respond to users' language patterns and preferences, paving the way for more natural and meaningful interactions in the realm of conversational AI.

References:

- ML eXchange (MLX) Examples Repository. Retrieved from: https://github.com/ml-explore/mlx-examples
- Mistral 7b working mechanism
- https://ml-explore.github.io/mlx/build/html/index.html