Chatbot Performance

Building a Financial Information Chatbot with LSTM and Transformers

This project explored the design and implementation of a domain-specific chatbot focused on financial terminology and education. The development process was divided into two phases: first, I constructed a custom sequence-to-sequence (Seq2Seq) chatbot using Long Short-Term Memory (LSTM) units enhanced with attention (using lab26 script as base), and second, the transition to a transformer-based model using Google's small model (500m parameters), the FLAN-T5.

Data Collection and Preparation

The dataset for training the chatbot was compiled from two main sources. I first developed a web scraper that crawls and extract financial definitions and explanations from the Investopedia website. Each entry was parsed into a standardized question-and-answer format, where the "input" field contained the financial term or query, and the "output" field provided the corresponding answer or explanation. In the project tree, the "raw" data files can be seen in each individual markdown file in the `./investopedia_terms` folder. This dataset was then serialized into a JSON file to facilitate preprocessing and model training.

In addition to the Investopedia data, a second dataset was created by manually extracting around 100 financial terms from a PDF glossary published by Creative Capital (can be found here - `./input/creative_capital_investment_term_glossary.pdf`. Each term was reviewed and paired with its definition in a similar Q&A format, then added to the existing JSON dataset. This manual augmentation ensured the inclusion of diverse financial topics, especially those that are considered essential

LSTM-Based Seq2Seq Model with Attention

The initial chatbot model was built using an encoder-decoder architecture with LSTM layers. The dataset consisted of JSON-formatted pairs of questions and answers focused on financial concepts (e.g., "What is an asset?", "What is risk?"). During preprocessing, each answer was modified to include a special `STARTSEQ` token and appended with an `ENDSEQ` token to indicate the start and end of the target sequence during training. Tokenization was performed using Keras' `Tokenizer`, and padding with zeroes ensured that all sequences had consistent length.

The encoder converted the tokenized input question into a fixed-length context vector which the decoder used to generate the output sequence token by token. In the beginning, the decoder was unable to effectively learn long-term dependencies which lead to repetitive or generic outputs. I implemented an attention mechanism manually within the decoder architecture to try to fix this issue. The attention mechanism allowed the decoder to selectively grab the relevant parts of the input sequence at each decoding step. The attention weights were then used to compute a context vector that was concatenated with the decoder's output before the final prediction.

Even though there were improvements in coherence and relevance from the addition of attention, training was still sensitive to hyperparameters. I introduced dropout layers to prevent overfitting and applied early stopping based on validation loss. Learning rate tuning further helped stabilize training and I slowly included all of the data (only trained on 50% of data as I adjusted batch size, epochs, etc. as limitations to GPU arose).

Inference was handled by saving and reloading the encoder and decoder models separately which allowed for step-by-step generation of outputs. I built an interactive interface using `ipywidgets` in Jupyter Notebook which enabled real-time demonstration of the model’s capabilities. Unfortunately, the LSTM model, even with attention, did not perform well.

Transformer-Based FLAN-T5

Although the LSTM model demonstrated the foundational principles of attention and sequence modeling, its limitations became evident in terms of fluency and factual responses (too much gibberish and or incorrect answers). I transitioned to a pretrained transformer model to try to overcome these issues. After some research in HuggingFace forums, I landed on using google's small model `flan-t5-base` which is perferct for my Q/A data format as it is a variant of T5, which takes text as input and output. Transformers apply self-attention across all token positions, so this should make them far more effective for modeling complex language tasks.

To fine-tune FLAN-T5 on the dataset, I reformatted each input as an instruction, rather than just a question such as "what is", by using the prefix "question:". This format aligns with the model's pretraining data, which focuses on instruction following. The tokenizer converted the data into token IDs and padding was handled with care to ensure that target labels used `-100` instead of the padding token ID to properly mask out ignored positions during loss computation. I used Hugging Face's `Seq2SeqTrainer` along with `DataCollatorForSeq2Seq` to streamline the training process, manage padding dynamically, and support label masking.

One important thing to note was that all training was performed on my personal Mac M4 system with an Apple Silicon chip. This made training more nuanced since I had to research and switch over to using the Metal Performance Shaders (MPS) backend of PyTorch to enable GPU training.

To evaluate the chatbot’s performance, I implemented a ROUGE-L scoring mechanism. On a representative subset of the dataset (20 samples), the fine-tuned FLAN-T5 model achieved an average ROUGE-L F1 score of approximately **0.42**, indicating a moderate level of overlap between generated answers and true responses. This score suggests that while the model generates coherent and relevant responses, further improvement, specifically in factual precision and paraphrase variety, could enhance alignment with reference answers. Further work could also lead to enhancing the dataset to present the same questions in a different way, for example, for a record like "what is an asset?" I could also have something like "define an asset for a company".

Conclusion

The development of this chatbot illustrates a clear progression from traditional neural architectures toward transformer models. The LSTM-based Seq2Seq model provided a good baseline for the chatbot in understanding sequence generation and attention. However, its limitations in handling long sequences and generating facts led me to transformer architectures.

The FLAN-T5 transformer demonstrated superior performance in language generation (it finally answered basic questions correctly!). Another thing to consider was the successful use of Apple Silicon MPS for training and inference show that modern workflows can be developed effectively on consumer-grade hardware - especially fine tuning models such as this. Future work would include expanding dataset, re-phrasing questions and/or answers, and fine-tuning a larger transformer model (>500m parameters).