**COMP9444 Project Summary**

Rick and Morty Chatbot - A scalable LLM based chatbot

1. **Introduction**

The primary goal of this project was to develop a character chatbot that emulates the unique speaking style of Rick from the popular TV sitcom “Rick and Morty”, by drawing on the recent evolutions in the space of Natural Language Processing (NLP) through Large Language Models (LLMs).

The ability to generate text in a specific character’s style has vast applications in entertainment, virtual assistance, and interactive storytelling. For instance:

1) Virtual influencers who can interact with social media followers to promote products, share contents, and therefore offer a fresh approach to marketing;

2) Customized chatbot by training the model with conversational data from previous customer interactions; and

3) Interactive/adaptive learning chatbot so that students can engage with virtual characters acting as intelligent tutors and receive personalized feedback.

**Literature Review**

We were inspired by a similar project (Rostyslav 2020), but with outdated code and references that could not be directly used for our purposes. We first started with replacing old features and libraries and restructuring the codebase. Through joint project work, we were able to build our own structure of preprocessing dataset, load/fine-tuning models, evaluating models, and obtain the final working chatbot.

1. **Methods**

Our project utilised Microsoft’s DialoGPT-small LLM as a starting point. We then undertook steps to enhance this model with additional training data from Rick and Morty’s dialogue script data. The DialoGPT is a transformer based pre-trained model that is open sourced and available through the Transformers library in python. The model utilises a modified version of the Transformer which only includes the decoder part of the original Transformer model.

Our modelling setup allowed us to vary the data input, vary model parameters (as further described below) and conduct model evaluation across a range of metrics specific to NLP type problems.

Once the model had been trained, we were able to interact with it and generate lines of output in the form of a chatbot. These were analysed both from a qualitative perspective (considering coherence, relevance, personality, robustness and context awareness), and a quantitative perspective (considering perplexity and additional metrics such as BLEU and METEOR).

1. **Experimental Setup**

**Input data and data processing steps**

Our original dataset was scraped as part of a Kaggle project (Andrada 2020) for the purposes of sentimental analysis. We utlised this for our own modelling purposes. The data was set up with each record containing a line of dialogue, and contextual (historical) dialogue leading up to a particular line of dialogue serving as features. The Project Notebook summaries both raw scraped data and the data setup for modelling.

The dialogue data was time-ordered and was preprocessed to be appropriate for training/validation/test and by utilizing the Pytorch data class. Below were the key data processing steps taken to prepare the data for modelling purposes:

• Adjusted block size with the models’ maximum sequence and single sentence length from dataset so that the model and tokenizer could handle the input text.

• Created context-response pairs (previous dialogue as features and current as the label) with the previous 7 lines so that it can be used for conversation context.

• Reversed/encoded/flattened text data into a format that the model could understand using corresponding tokenizer of the model.

• Loaded and cached preprocessed/tokenized dataset for each purpose, 90% of the data used for training and 10% for validation/test. We made sure that the evaluation data is not randomised but only training data to ensure reproducibility of results.

We also undertook exploratory data analysis to better understand the type of data, and present our results in the Project Notebook. The style of Rick’s speech was clearly visible in the EDA, and helped us in future modelling stages and model evaluation.

**Model Training**

We then undertook fine-tuning of pre-trained language model as follows:

* Used Tensorboard writer to visualize and monitor training process/results, and handled padding of the input sequences to make sure they have the same length.
* The model’s token embeddings were then resized to match the tokenizer’s vocabulary size
* Set up the optimizer (AdamW), learning rate scheduler (with or without weight decay), gradient clipping, and gradient accumulation. The Python logging library was used to log/print information about the training.
* Iterated over all epochs and batches, and processed/backpropagated to update model’s parameters. The model’s checkpoints were saved for every 300 steps (about 2/3 of each epoch).

**Evaluation Strategy**

1. NLP specific evaluation metrics:

We utilised BLEU, METEOR, ROUGE metrics using NLTK library to test the model performance (Caglayan et al. 2020). Prepared sets of example prompt found from recent seasons of sitcom, generated text and compared with test dataset using these scores. Observed inconsistent results from ROUGE metrics so that excluded for further experiments.

1. Human evaluation:

We undertook a high level human evaluation with multiple outputs (Turing-like test). We prepared sets of example prompt found from recent seasons of sitcom, generated text with our model/different models along with actual script and ranked their similarity. The human evaluation took a relatively simple form of judgement based assessment, rather than a more formal double-blind type of evaluation (given time constraints).

1. Evaluation across an alternative dataset:

Implemented the same approach to a dataset containing multiple quotes from ancient philosophers (Chris 2020) to make sure this approach is effective, and also works in different circumstances.

**Rationale**: We tested multiple models for this task such as GPT2, DialoGPT-small, DialoGPT-large, Pygmalion-6B, and confirmed that DialoGPT-small is indeed most appropriate for our project considering the performance in terms of robustness in capturing conversational nuances, and size of the model because of the limited time/computing resources. The results of our tests are captured in our notebook.

**Parameter adjustments**: We further tested to find the best model by changing training epochs, learning rate, context size, size of the model, optimizer, block size, and combination of these parameters followed by thorough testing with above methods to check the performance improvement and prevention of overfitting. Our results are again captured in the notebook.

**Hyperparameter testing**

In developing our hyperparameters, we looked across a range of dimensions:

* Experiment of how the metrics change over different parameters and models.
* Combination of multiple variations to find the best performance model.
* Human evaluation with prompt text (not in our dataset to avoid target exposure), prepare 4 different outputs from our model, another models, and actual script to check if we can figure out which is which.

**We undertook the following key hyperparameter tests:**

1. Test by changing models (DialoGPT-small, DialoGPT-large, GPT2, Pygmalion-6B)
2. Test by changing the number of previous responses to consider (5 ~ 24)
3. Test by changing training epochs (1 ~ 9)
4. Test by changing learning rate (0.00001 ~ 0.0001)
5. Test by changing block size (128 ~ 1024)
6. Test by changing optimizer (AdamW, SGD)
7. Test by grouping changes above

The results for these are briefly discussed below, and in more detail in our Project Notebook.

1. **Results**

**Experimental results:**

1. Larger models tend to produce slight better perplexity with the same amount of training. However, also required more memory and training time to reach the acceptable level. This could be because our task is less about understanding complex language structures and more about learning a new style, which can be achieved through effective fine-tuning. Therefore, DialoGPT-small was found most efficient.
2. Changing the size of context led to most dramatic improvement of the model in terms of perplexity while also increasing training time. For some experiments, we were able to produce perplexity scores of close to 1, which is arguably indicates overfitting, since it means the model is almost always able the next token with certainty that is too good to be true.
3. Changing training epochs also helped to increase the model performance. However, considering our dataset size is relatively small, we observed that any number more than 8 epochs would lead to overfitting. Any number lower than 2 observed to underfit our model and produced extreme values of metrics.
4. Changing learning rate helped the model to reach the accepted level faster, but also produced inconsistent result as sometimes it failed to learn the style of Rick, checked by metrics. Any number lower than 0.00003 observed to underfit our model.
5. Changing block size, optimizer did not affect to the performance. It is possibly because our task was more dependent on the model’s ability to capture the unique dialogue pattern of Rick, which is primarily influenced by the fine-tuning process. For the imitation task like ours, majority of the learning has already occurred during its pre-training phase, and during the fine-tuning, we are essentially nudging the model’s parameters to better suite with our specific task.

By Grouping above findings, we could build to best performance model (10 context size, 7 training epochs, 0.0001 learning rate) and observed 1.8868 Perplexity, 0.13 and 0.44 for BLEU and METEOR.

**Reflections on the metrics used**: ROUGE produced inconsistent results and therefore, we did not use it for our experiments. We believe that the primary reason is the differing emphasis on precision and recall. BLEU and METEOR give a balanced view on both precision and recall (Chris 2020), but ROUGE focuses more on recall so that better for evaluating text summarization and translation task. Because a language is highly dependent on context, it is likely that with more data, the model can expect what will be next easier. However, this does not necessarily mean the quality of generated text also increase. Moreover, considering the character we are creating is a cynical mad scientist, evaluation metrics might not be able to capture the performance. Because of this limitation, we added human evaluation at the end and tried the same approach with another dataset.

**Human evaluation/comparison with state of the art models**: We generated responses from our model and other models (GPT4 with chatGPT, PaLM2 with Google Bard), along with actual scripts, and asked each member to rank then based on how similar they were to Rick’s dialogue style. Feedback was generally positive, with our model’s outputs often being ranked highly so that it seems like our model is good enough to be deployed and used as Rick chatbot.

1. **Conclusions**

**Contributions**: We developed a solid understanding of working with NLP models, and the ever growing Large Language Model echo systems. The course material provided necessary knowledge and confidence to work with advanced model pipelines and complex tasks. Moreover, this project was a valuable chance to practice time management when working through a complex task.

**Strengths**: The model's key strength lies in its ability to capture the essence of Rick's speech style. Through careful selection of model architecture and data preprocessing, the system demonstrates an impressive proficiency in mimicking Rick's idiosyncratic expressions and mannerisms. This not only enriches the generated content but also opens doors for applications in entertainment and media where character-specific content generation is essential.

**Limitations**: Despite the encouraging outcomes, the project also unveils certain limitations. The model's performance has shown to be sensitive to hyperparameter adjustments, indicating a narrow optimal range for parameters like learning rate and training epochs. Additionally, the emulation of Rick's speech, although promising, requires further fine-tuning and experimentation to reach a level where it could be considered indistinguishable from the actual character's speech.

**Future Work**: As a natural extension of this work, several avenues for future exploration present themselves. One we can do is exploring more models/datasets (Llama2, other characters), different preprocessing techniques, which could help to better capture the unique vocabulary and style. Furthermore, we may dive deeper into the cost function and additional machine learning techniques such as Reinforcement Learning with Human Feedback.

References

Rostyslav Neskorozhenyi, 2020, Make your own Rick Sanchez (bot) with Transformers and DialoGPT fine-tuning, Towards Data Science, accessed 10 Jul 2023, < <https://towardsdatascience.com/make-your-own-rick-sanchez-bot-with-transformers-and-dialogpt-fine-tuning-f85e6d1f4e30>>.

Andrada Vulpe, 2020, Sentiment Analysis: Rick and Morty Scripts, Kaggle, accessed 10 Jul 2023, < <https://www.kaggle.com/code/andradaolteanu/sentiment-analysis-rick-and-morty-scripts/notebook>>.

Ozan Caglayan, Pranava Madhyastha, Lucia Specia, 2020, Curious Case of Language Generation Evaluation Metrics: A Cautionary Tale, Proceedings of the 28th International Conference on Computational Linguistics, pages 2322–2328

Chris Lemke, 2020, Philosophical Texts, Kaggle, accessed 21 Jul 2023, < <https://www.kaggle.com/datasets/christopherlemke/philosophical-texts?resource=download&select=sentences.csv>>.