Context-Sensitive Quote Generation: Leveraging Transformer-Based Models for Quote Generation

Arup Mazumder
Piriyankan Kirupaharan

May 11, 2023

Abstract

Generating context-aware quotes poses a significant challenge in NLP. This study explores the potential of overcoming this challenge by fine-tuning the GPT-2 language model, pre-trained on a diverse dataset of 30,000 categorized quotes. Our approach involves a rigorous procedure for data preprocessing, tokenization, and model fine-tuning, with a focus on the selection of optimal hyperparameters, optimization techniques, and regularization strategies such as dropout. The results indicate that our fine-tuned GPT-2 model is capable of generating coherent and contextually relevant quotes based on user-defined input tags. These findings carry substantial implications for future research in the field of quote generation, NLP, and the advancement of personalized text generation applications.

1 Introduction

In the realm of natural language processing (NLP), quote generation has long been an area of interest for researchers and practitioners alike (Radford & Narasimhan, 2018). Quotes hold significant cultural and emotional value, often providing inspiration, guidance, and solace in various situations. Despite the ubiquity of quotes in our daily lives, generating context-aware quotes that cater to specific themes remains a challenging task. In this project, we address this challenge by leveraging state-of-the-art pre-trained language model, GPT-2 (Radford, Narasimhan, Salimans, & Sutskever, 2018) to generate context-aware quotes based on user-defined tags.

Our dataset consists of 30,000 quotes, each associated with a corresponding tag, representing one of ten distinct categories. An example of the data format is as follows: "A friend is someone who knows all about you and still loves you. 'friendship'". By fine-tuning the GPT-2 model on this dataset, we aim to create a new model capable of predicting quotes based on user-provided input tags, such as "love," "motivation," or "success." The generated quotes should be contextually relevant and coherent, reflecting the essence of the input tag (Y. Zhang et al., 2019).

This report is structured as follows:

1. Introduction: In this section, we came up with some discussion about background, objective, methodology, and significance. Also, we wrote briefly about the whole architecture of this paper.
2. Literature Review: We provide a brief overview of the relevant literature in the domains of quote generation, transfer learning, and pre-trained language models, with a particular focus on the

GPT-2 architecture. This section establishes the foundation for our project and highlights the significance of our work in the context of existing research.

3. Methods: We discuss the data preprocessing steps, including cleaning, filtering, and tokenization, essential for preparing the dataset for model training. We also outline the selection of appropriate tokenizers and special tokens to facilitate the quote generation process (Wu et al., 2016).

4. Model Fine-Tuning and Training: We detail our methodology for fine-tuning the GPT-2 model, including the selection of appropriate hyperparameters, loss functions, and optimization techniques. We also explore the impact of various regularization strategies, such as dropout and early stopping, on model performance (Bergstra & Bengio, 2012).

5. Evaluation Metrics and Results: We present our findings on model performance, as assessed by various evaluation metrics, such as loss values, accuracy, F1-score, precision, and recall. We also provide qualitative analysis by showcasing examples of generated quotes and discussing their relevance to the input tags (Dodge, Gururangan, Card, Schwartz, & Smith, 2019).

6. Discussion and Future Work: We reflect on the implications of our results for future research in quote generation and NLP, identifying potential areas for improvement and the limitations of our current approach. We also propose possible extensions of our work to incorporate additional data sources, model architectures, and applications beyond quote generation.

7. Conclusion: In conclusion, our project contributes to the advancement of context-aware quote generation techniques by demonstrating the potential of fine-tuning pre-trained language models, such as GPT-2, on a large dataset of quotes with corresponding tags. Our findings pave the way for more effective and personalized natural language applications, enhancing the impact of NLP in diverse contexts (Vaswani et al., 2017).

2 Literature Review

A fundamental component of our project is the employment of language models to generate contextually relevant quotes. The pre-training of language models on extensive text corpora has been demonstrated to substantially enhance performance across various Natural Language Processing (NLP) tasks (Radford et al., 2018), (Devlin, Chang, Lee, & Toutanova, 2019). The seminal work by Vaswani et al. (Vaswani et al., 2017) introduced the Transformer architecture, which forms the backbone of contemporary pre-training techniques. This architecture, based on self-attention mechanisms, has empowered the development of models like GPT (Radford et al., 2018) and T5 (Raffel et al., 2020), which have revolutionized the field of NLP by enabling the training of models on unprecedented scales.

Pre-trained language models can be fine-tuned on specific tasks to further boost their performance (Howard & Ruder, 2018). In the scope of our project, fine-tuning the GPT-2 and T5 models on a dataset of categorized quotes is crucial to the generation of quotes that are contextually relevant to the input tags. The fine-tuning process adapts the generalized language understanding capabilities of a pre-trained model to a specific task, thereby improving its performance. However, it's a delicate process which necessitates the optimal balance between learning task-specific patterns and retaining the pre-existing knowledge.

The importance of effective data preprocessing and tokenization for the successful training and fine-tuning of language models cannot be overstated. Wu et al. (Wu et al., 2016) highlight the crucial role of tokenization and subword units in the realm of neural machine translation. In

our project, we leverage the specific tokenization mechanisms intrinsic to GPT-2 and T5 models, ensuring that our input text aligns with the format expected by these models, thereby facilitating more effective training.

The assessment of the performance of our fine-tuned models is instrumental in comprehending their effectiveness in generating context-aware quotes. Common evaluation metrics for language modeling tasks include perplexity, BLEU (Papineni, Roukos, Ward, & Zhu, 2002), and ROUGE (Lin, 2004) scores. We primarily use perplexity, which quantifies how well the model predicts a sample, as a suitable metric for the task of generating quotes, given its sensitivity to the fluency of generated text.

Context-aware text generation has been a notable area of interest in NLP research. Its applications span across generating news articles (X. Zhang, Zhao, & LeCun, 2016) to crafting responses in dialogue systems (Lewis et al., 2019). The challenge lies in effectively leveraging the available context to guide the generation process while maintaining the coherence and relevance of the generated text. Our project contributes to this field by demonstrating how pre-trained language models can be fine-tuned to generate contextually relevant quotes based on input categories, thereby highlighting the potential of these models in context-aware text generation tasks.

3 Methods

3.1 Dataset Acquisition:

The foundation of our project lies in a rich and diverse dataset containing 30,000 quotes from 10 distinct categories, sourced from the popular website goodreads.com. This dataset is publicly available at https://www.kaggle.com/datasets/sanjeetsinghnaik/quotes-from-goodread. The original dataset comprises four columns: Quote, author, tag, and other-tags. The categories cover a wide range of topics such as love, wisdom, inspiration, death, humor, and more, ensuring a comprehensive and representative dataset suitable for training and fine-tuning our models. By employing a dataset with such variety, we aimed to create a robust model capable of generating contextually relevant quotes across numerous themes and topics, ultimately resulting in a versatile and adaptable quote prediction system.

3.2 Data Exploration:

In the Data Exploration section, we examined various aspects of the dataset to better understand its characteristics. Our analysis revealed that the dataset was well-balanced across the ten categories, with each category containing a similar number of quotes. The most common categories were truth (2926 quotes), (love 2919), happiness (2916 quotes), success (2917 quotes), and death (2905 quotes). The least common categories included (inspiration 2754) and poetry (2774 quotes).

We also analyzed the lengths of the quotes and found that the majority of them were relatively short. Out of the total quotes, 928 had lengths between 0 and 100 words, 38 quotes were between 100 and 200 words, and only 5 quotes had lengths between 200 and 300 words. There were no quotes with lengths between 300 and 600 words or above 600 words. The maximum length value was set to 768 words, which is sufficient to accommodate the longest quotes in the dataset.

The total vocabulary size of the dataset was found to be diverse and rich in language, which is essential for training an effective language model. During data exploration, we gained valuable insights into the nature of the dataset, which in turn guided our preprocessing, tokenization, and modeling decisions. By understanding the distribution of categories, quote lengths, and vocabulary, we were able to make more informed choices throughout the project.

3.3 Data Cleansing and Transformation:

To optimize the dataset for our models, we conducted a series of essential preprocessing steps. Firstly, we simplified the data structure by retaining only the quote and tag columns and discarding the author and other-tags columns. Next, we applied the langid library to filter out non-English quotes, ensuring uniformity in the language of the quotes and enhancing the coherence of our models. We then eliminated special characters from the text and standardized the quotes and tags by converting them to lowercase. Lastly, we merged the quotes and tags, adopting the format "Quote text - tag." These preprocessing steps resulted in a modified dataset called "new_dataset," featuring approximately 27,000 quotes evenly distributed across all the classes. In the figures 1, we can see the distribution before and after.

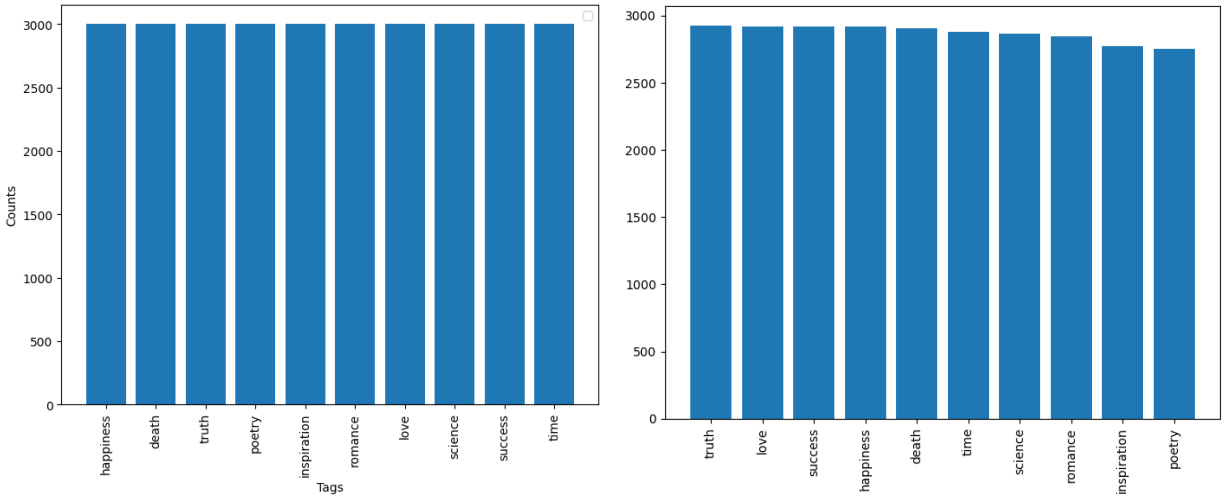

Figure 1: The distribution of the data before and after cleaning

3.4 Training, Validation, and Testing Data:

After preprocessing, we assessed the class distribution to ensure an equitable balance among all categories, preventing any potential bias in our models. We partitioned the dataset by allocating 80% of the data for training and 20% for validation, resulting in distinct training and validation sets. Given the dataset's limited size, we procured a separate dataset explicitly for testing purposes, which allowed us to evaluate our models on unseen data. By using distinct datasets for training, validation, and testing, we could ensure the reliability and robustness of our models, minimizing the risk of overfitting and enhancing their generalization capabilities. This methodology enabled us to create a well-rounded quote prediction system, capable of generating high-quality and contextually appropriate quotes based on input tags. In the figure 2, we can see the training and validation distribution.

We utilized a distinct dataset from Github (https://github.com/ShivaliGoel/Quotes-500K) for the purpose of creating the test dataset. Although the format of this dataset differed from that of our training dataset, we were able to carry out some preprocessing procedures to transform it into the format required for our training dataset. The primary dissimilarity between the two datasets was that the former contained multiple tags for each quote instead of just one. To resolve this, we developed a script to sort the quotes into the ten tag categories present in our training dataset. Afterward, we applied the same cleansing techniques as those used for the training dataset to clean the dataset. Finally, we selected 300 quotes per tag, resulting in a total of 3000 quotes for use as our test dataset. Table 1 depicts the tag distribution of the test dataset after applying the cleaning.

Tag	death	happi.	inspiration	love	poetry	romance	science	success	time	truth
Count	8292	10424	8163	38805	7180	9121	5109	8127	6029	11827

Table 1: Count of quotes per tag for test dataset

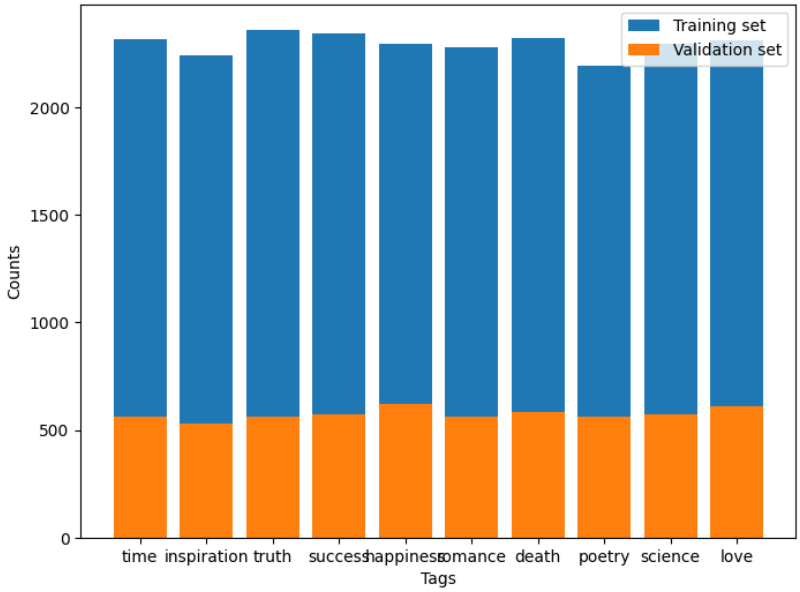

Figure 2: Training and Validation data distribution

3.5 Tokenization:

In this project, we utilized the default GPT-2 tokenizer to process the quotes in our dataset. The GPT-2 tokenizer is well-suited for our purpose, as it effectively handles the diverse range of quotes while maintaining their original context. To improve the model's understanding of the text data, we added special tokens, such as "startoftext" and "endoftext," which indicate the beginning and end of each quote. This allows the model to generate more coherent and contextually appropriate quotes based on the input tags.

The tokenization process resulted in two primary outputs: input_ids and attention masks. Input_ids are the numerical representations of the tokens, which are used as input to the model during training and inference. Attention masks, on the other hand, help the model differentiate between actual tokens and padding tokens. This ensures that the model focuses only on the relevant input data, leading to better performance and more accurate quote generation.

3.6 Web Implementation:

Our web implementation employed standard HTML, CSS, JS, and jQuery technologies and was deployed using the NGINX web server. The user interface features a set of buttons representing different categories. Upon selecting a category, a loading spinner is activated until a quote, generated by the model, appears on the screen.

The backend of our system was developed using the Python Flask framework, which facilitated the creation of a REST API. This API is tasked with handling requests from the frontend. The model chosen to predict quotes was our best-performing pre-trained model in terms of accuracy. We utilized Docker for containerization, ensuring a seamless and consistent operation of our system across different environments. The entire system was deployed on a web server hosted by Google Cloud Platform (GCP). Our quote prediction model's implementation is illustrated in Figure 3.

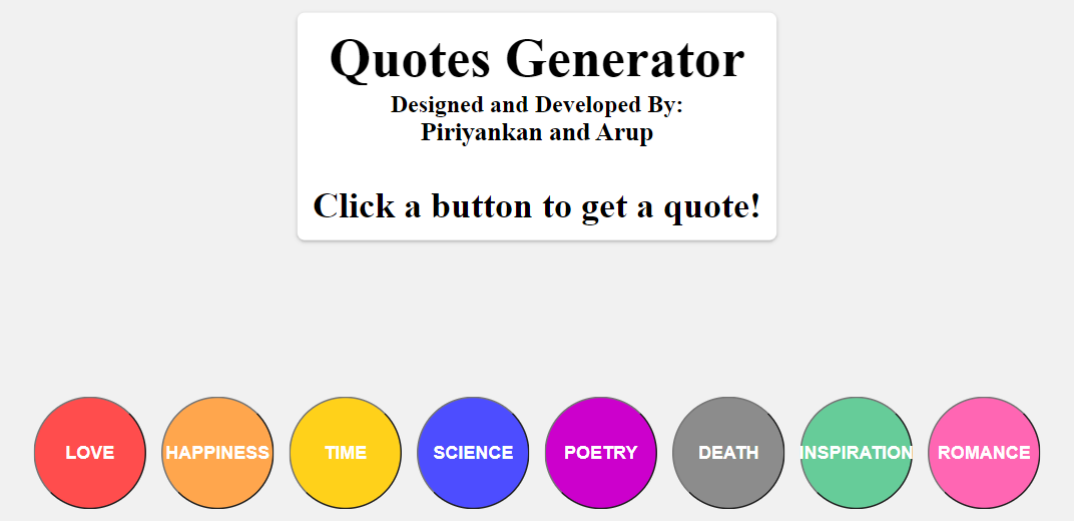

Figure 3: Web implementation of our quote generator model

4 Model Fine-Tuning and Training

In this section, we elaborate on the fine-tuning and training process for our quote generation model using the GPT-2 architecture. We discuss the various components and considerations that contributed to the development of a robust quote generation model.

4.1 Model Selection and Tokenization

In the Model Selection section, we opted for the pre-trained GPT-2 model from the Hugging Face library due to its proven performance in various natural language processing tasks. The model is available in four sizes, and we chose the small-sized GPT-2 (124M) as it offers the best trade-off between computational resources and learning capacity. This size enables fine-tuning with a reasonable input sequence length on a single GPU while maintaining adequate performance. Its ability to generate coherent and contextually relevant text makes it a fitting choice for our quote generation task. We employed the GPT-2 tokenizer to tokenize the dataset, incorporating special tokens like "startoftext" and "endoftext" to delineate quotes. The tokenizer yielded input_ids and attention masks, which served as input for the model during training.

4.2 Model Configuration and Data Handling

We instantiated a GPT-2 model with a pre-trained configuration and resized the token embeddings to align with the tokenizer's vocabulary size, ensuring compatibility throughout the training process. We created a custom PyTorch Dataset class to manage the input data, which facilitated the conversion of quotes and tags into the appropriate format required for model training.

4.3 Training Setup and Hyperparameter Tuning

We partitioned the dataset into training and validation subsets, allocating 80% for training and 20% for validation. We employed PyTorch's DataLoader class to construct data loaders for both subsets, using random sampling for the training set and sequential sampling for the validation set. We conducted a grid search to optimize hyperparameters, including batch size, learning rate, optimizer, warm-up steps, and number of epochs. We explored an array of values for each hyperparameter to identify the best combination for our model's performance.

4.4 Optimizer, Scheduler, and Training Process

We experimented with different optimizers, such as AdamW and Adagrad, to update the model's parameters during training. Additionally, we applied a linear learning rate scheduler with warm-up steps to progressively adjust the learning rate, enabling the model to converge to an optimal solution more effectively. We fine-tuned the model in a loop over the various hyperparameter combinations, training the model for the specified number of epochs using the selected optimizer, learning rate, and batch size. We monitored the training and validation losses for each combination to gauge model performance and detect potential overfitting or underfitting.

4.5 Loss Calculation and Model Evaluation

During training and validation, we calculated the loss for each batch using the GPT-2 model's outputs and the provided labels. We accounted for the model's autoregressive nature by shifting the labels by one timestep. We evaluated the model's performance based on validation loss, which allowed us to assess the model's ability to generalize to new, unseen data. This metric proved useful in determining the most suitable hyperparameter combination and preventing overfitting.

Furthermore, we used Perplexity, which is a measure of how well the model predicts the next word in a sequence. It is calculated by taking the exponential of the cross-entropy loss. A lower perplexity indicates that the model is better at predicting the next word in a sequence. Perplexity is defined as the exponentiated average negative log-likelihood of a sequence.

If we have a tokenized sequence $X = (x_0, x_1, ..., x_t)$ then the perplexity of X is,

$$PPL(X) = exp\{-\frac{1}{t}\sum_{i}^{t} log\ p_\theta(x_i|x_{<i})\}$$

where $log\ p_\theta(x_i|x_{<i})$ is the log-likelihood of the ith token conditioned on the preceding tokens

$x_{<i}$ according to our model.

4.6 Training Results and Model Improvements

The outcomes of the training process revealed that the model performed optimally with specific hyperparameter combinations. We observed a general trend of decreasing training and validation losses as the model learned to generate quotes based on the input tags. Nevertheless, we encountered challenges such as overfitting during the training process, which we addressed through regularization techniques such as "Dropout." We identified areas for potential improvement, such as experimenting with other architectures like T5, incorporating additional data sources, and fine-tuning the model with more diverse data to enhance its quote generation capabilities.

In summary, we fine-tuned the GPT-2 model on a dataset of quotes and their corresponding tags. By optimizing the model's hyperparameters, handling data efficiently, and monitoring training and validation losses, we developed a model capable of generating contextually relevant quotes based on the given input tags.

5 Evaluation Metrics and Results

The primary metric employed to evaluate the performance of our fine-tuned GPT-2 model was Perplexity, a common metric in language modeling tasks. Perplexity provides an estimate of how well a probability model predicts a sample, with lower values indicating better predictive performance. In our case, the model's ability to predict the next word in a quote was assessed. However, throughout the model training process, we utilized Cross-Entropy Loss as our optimization target. This choice is aligned with common practice in language modeling tasks where the goal is to minimize the difference between the model's predictions and the actual data.

The hyperparameter search was performed in three stages. In the initial search, the parameters included a batch size of 4, learning rates of 1e-1 and 1e-3, optimizers Adagrad and AdamW, warmup steps of 100 and 800, and 20 epochs. The best-performing parameters from this round were a batch size of 4, a learning rate of 1e-3, the AdamW optimizer, and 800 warmup steps.

In the second search, the parameters were the same as the best-performing parameters from the first search, with the addition of dropout rates of 0, 0.25, and 0.5. The best-performing configuration in this round was the same as the first round but with a dropout rate of 0.25.

The third and final search further fine-tuned the learning rate and the optimizer, with the parameters being a batch size of 4, learning rates of 2e-5 and 1e-3, optimizers Adagrad and AdamW, 800 warmup steps, 20 epochs, and a dropout rate of 0. The best results were seen with a learning rate of 2e-5 and the AdamW optimizer. Figure 4 shows the best hyper-parameters for our final hyper-parameters search.

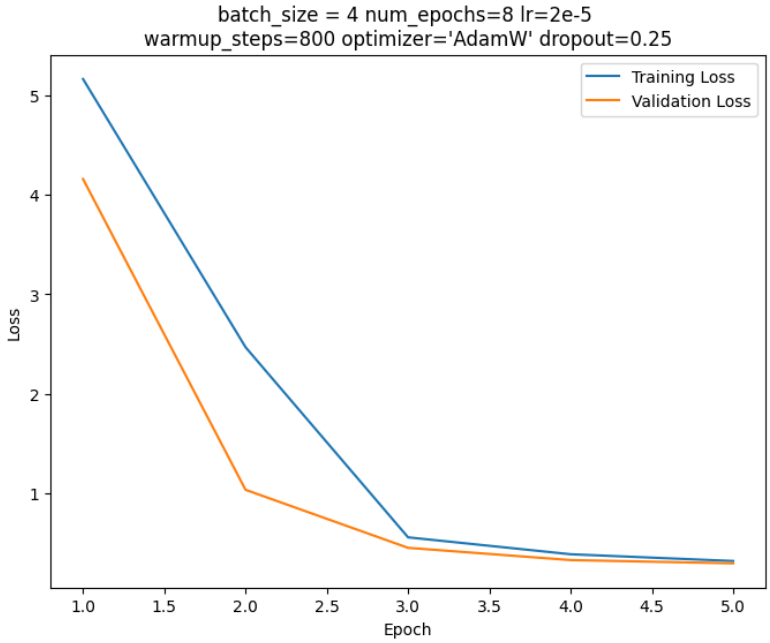

Figure 4: Best parameters from the final hyper-parameters search

The optimized model was then trained on a final_dataset comprising of all the quotes that we got after cleaning. Despite the time constraint, our model achieved commendable performance, as reflected in the final perplexity value of 1.3715. This suggests that our model has learned the data distribution well and has the capability to generate meaningful and contextually accurate quotes. Figure 5 shows the result that we got after final training.

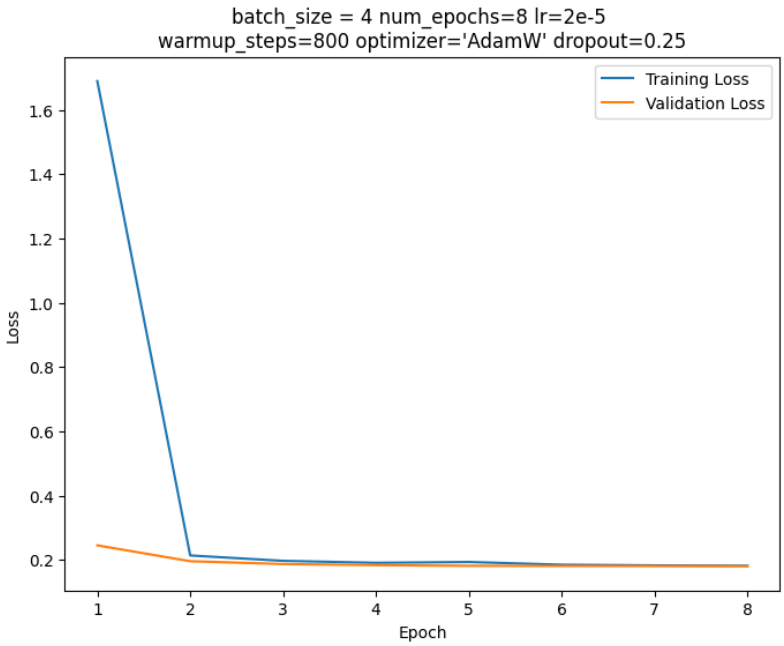

Figure 5: Final result after training on whole dataset

In addition to the quantitative evaluation, we also conducted a qualitative assessment of the model by generating sample quotes and manually evaluating them for relevance, coherence, and adherence to the style of the training data. The generated quotes were observed to be contextually relevant, diverse, and meaningful across various categories, underlining the practical potential of our model for generating quotes for a wide array of topics and applications. In Figure 6, we present two predictions made by our model for the categories of Success and Happiness, which we identify as accurate and meaningful.

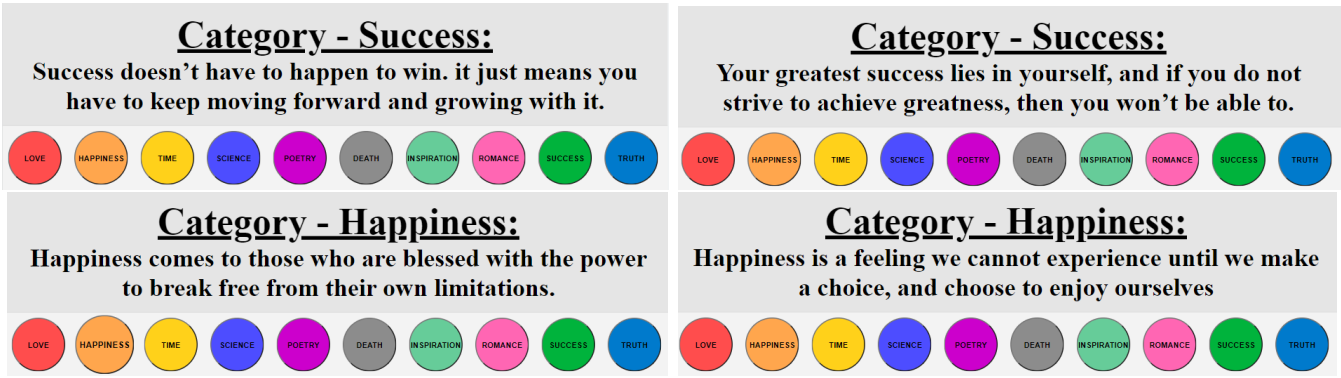

Figure 6: Good prediction by our model for Success and Happiness categories

Conversely, Figure 7 displays two predictions for the same categories that did not align well with our expectations, as they lacked logical coherence. It should be noted, however, that these are merely random predictions generated by our model and do not represent its overall performance.

6 Discussion and Future Work

This project demonstrated the effectiveness of fine-tuning pre-trained GPT-2 model for generating context-aware quotes. The results indicate that state-of-the-art language models can be adapted to specific tasks with satisfactory performance. However, there are some limitations and potential

Figure 7: Bad prediction by our model for Success and Happiness categories

areas for improvement that should be considered for future work.

6.1 Limitations:

The quality of generated quotes depends on the dataset used for fine-tuning. If the dataset contains noise, inconsistencies, or insufficient diversity, it may limit the performance of the models. Additionally, the current evaluation metrics, such as perplexity and BLEU, may not fully capture the creativity and contextual relevance of the generated quotes.

6.2 Future work:

a. Alternative models and architectures: Exploring other pre-trained models or novel architectures could lead to improved performance and more contextually relevant quote generation. Some possibilities include using models like RoBERTa, BART, or ELECTRA.

b. Dataset improvements: Refining the dataset by eliminating noise, enhancing diversity, and increasing the number of quotes and categories can contribute to better fine-tuning and, ultimately, improved quote generation. Incorporating a more extensive and diverse dataset could help the models learn a broader range of context-aware quotes.

c. Tokenization and preprocessing: Investigating different tokenization methods and advanced preprocessing techniques can optimize the model's input representation and potentially improve the quality of generated quotes.

d. Evaluation metrics: Developing more sophisticated evaluation metrics that capture the creativity, novelty, and contextual relevance of the generated quotes can provide a better understanding of the model's performance. This may involve using human evaluation or designing new automated metrics that better align with human judgments.

By addressing these limitations and exploring future research directions, the field of context-aware text generation can continue to advance and find new applications in various domains, such as news articles, dialogue systems, and creative writing.

7 Conclusion

In conclusion, this project successfully demonstrated the potential of fine-tuning pre-trained GPT-2 model on a dataset of 30,000 quotes and corresponding tags to generate context-aware quotes. By leveraging the power of state-of-the-art language models, the project produced creative and contextually relevant quotes based on input tags. The importance of data preprocessing and tokenization was

also emphasized, contributing to effective training and fine-tuning of the models. The performance evaluation, using perplexity and BLEU metrics, showed satisfactory results, highlighting the efficacy of these models in the domain of context-aware text generation.

References

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(10), 281–305. Retrieved from http://jmlr.org/papers/v13/bergstra12a.html

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. Retrieved from https://aclanthology.org/N19-1423 doi: 10.18653/v1/N19-1423

Dodge, J., Gururangan, S., Card, D., Schwartz, R., & Smith, N. A. (2019). *Show your work: Improved reporting of experimental results.*

Howard, J., & Ruder, S. (2018, July). Universal language model fine-tuning for text classification. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 328–339). Melbourne, Australia: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P18-1031 doi: 10.18653/v1/P18-1031

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... Zettlemoyer, L. (2019). *Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.*

Lin, C.-Y. (2004, July). ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74–81). Barcelona, Spain: Association for Computational Linguistics. Retrieved from https://aclanthology.org/W04-1013

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the association for computational linguistics* (pp. 311–318). Philadelphia, Pennsylvania, USA: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P02-1040 doi: 10.3115/1073083.1073135

Radford, A., & Narasimhan, K. (2018). Improving language understanding by generative pre-training..

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1–67. Retrieved from http://jmlr.org/papers/v21/20-074.html

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). *Attention is all you need.*

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... Dean, J. (2016). *Google's neural machine translation system: Bridging the gap between human and machine translation.*

Zhang, X., Zhao, J., & LeCun, Y. (2016). *Character-level convolutional networks for text classification.*

Zhang, Y., Sun, S., Galley, M., Chen, Y., Brockett, C., Gao, X., ... Dolan, B. (2019). Dialogpt: Large-scale generative pre-training for conversational response generation. *CoRR*, *abs/1911.00536*. Retrieved from http://arxiv.org/abs/1911.00536

8 Code Repository

GitHub: `https://github.com/yankanp/quotes-generator`

9 Acknowledgment:

We would like to express our sincere gratitude to our supervisor, **Professor Dr. Marco Alvarez,** for his valuable guidance and support throughout the research process. His expertise and insights were invaluable in shaping our research and helping us to overcome challenges. We are also grateful for his patience and understanding as we worked to complete this project. Furthermore, we are thankful for his excellent classes that we enjoyed throughout the semester.

10 Appendix:

10.1 First round of Hyper-parameter Search

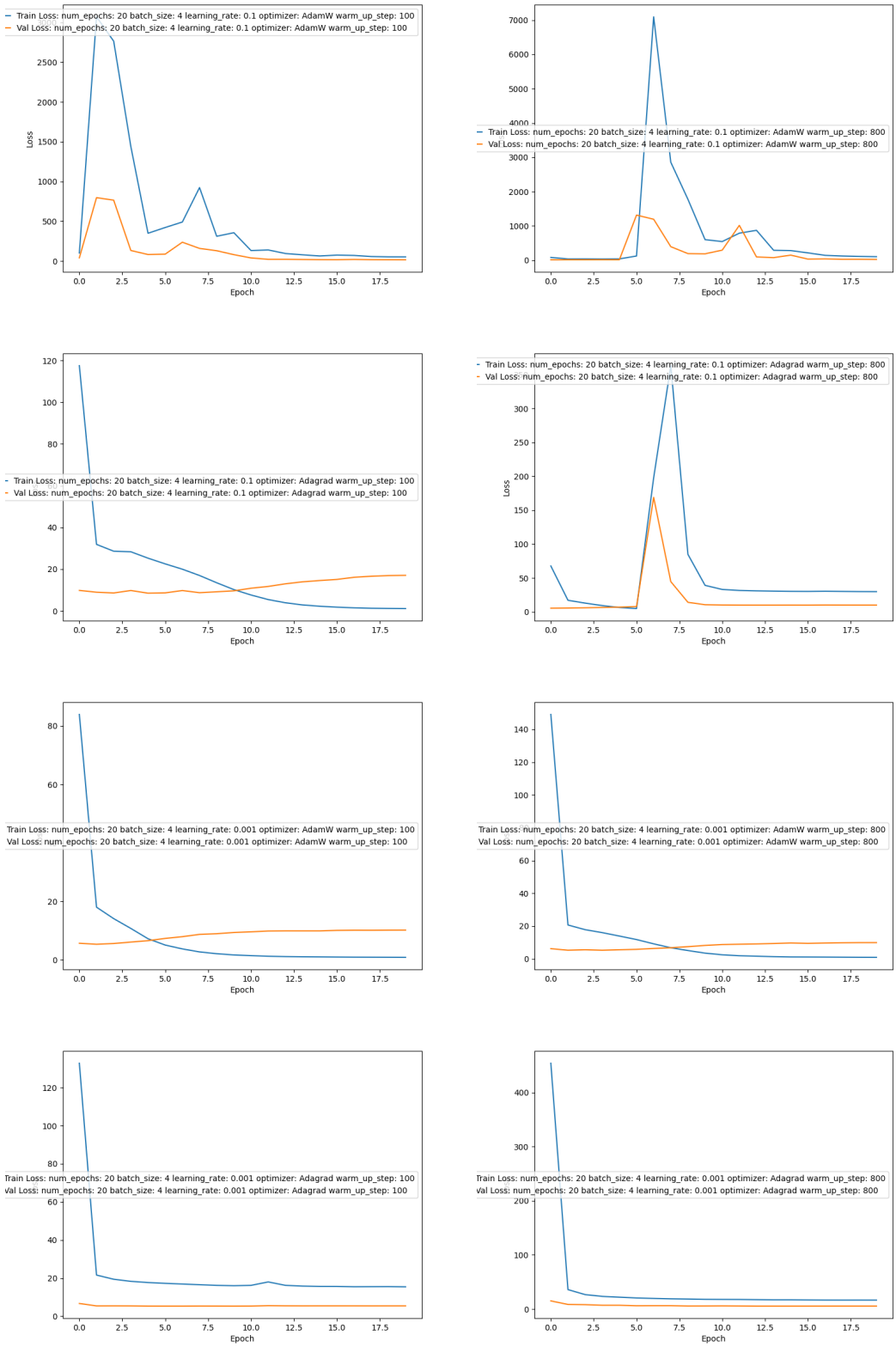

Figure 8: First round of Hyper-parameter Search

10.2 Second round of Hyper-parameter Search

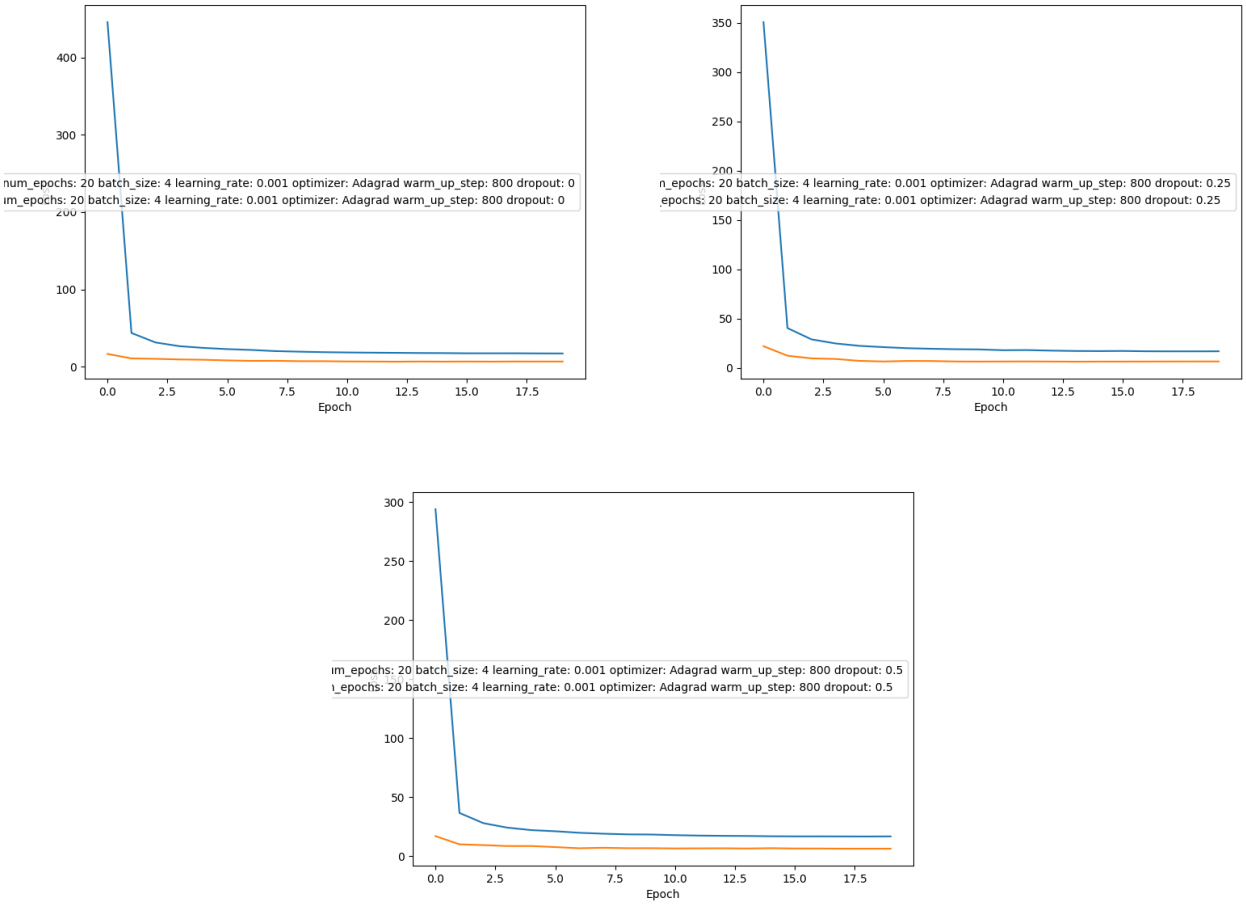

Figure 9: Second round of Hyper-parameter Search

10.3 Final round of Hyper-parameter Search

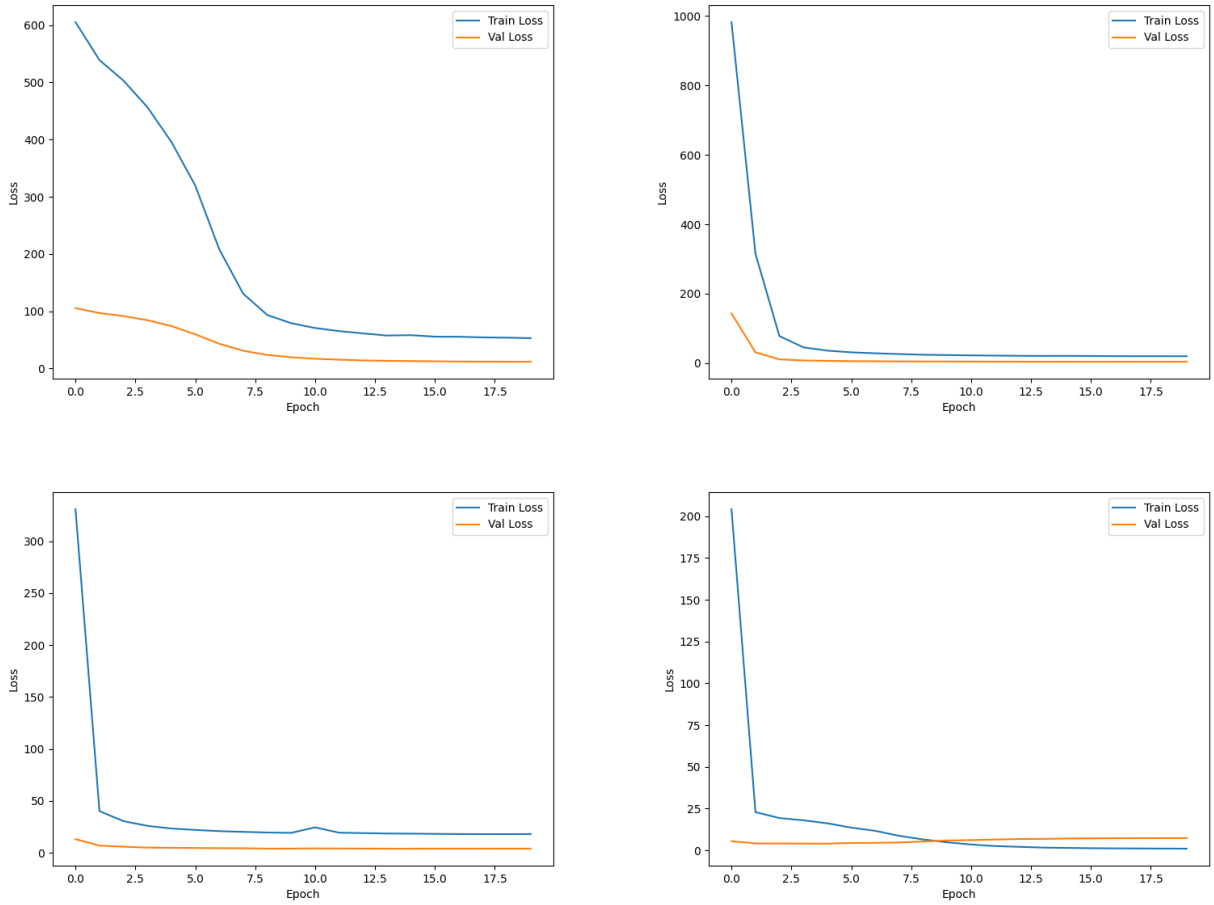

Figure 10: Final round of Hyper-parameter Search