CS 6120: Short-Story Generation using RNNs and GPT

Aakash Mahesha
Anusha Kalbande
Rishabh Raj

Abstract

This project uses natural language processing (NLP) strategies to produce short stories from fanfiction archives. Given the variety of stories, characters, locations, and themes they contain, fanfiction archives make a great source of inspiration for NLP projects on narrative development. The HuggingFace tools' Recurrent Neural Networks (RNNs) and Transformers are used as the models, and the HuggingFace API dataset is used as the main dataset. The effectiveness of the models is assessed manual assessment. Word clouds will be used to evaluate vocabulary and train-validate loss monitoring graphs and display the project's findings.

1 Introduction

For thousands of years, literature has played a crucial role in human society as a means of artistic expression. It includes a broad variety of literary forms, such as poetry, prose, short stories, and both fiction and non-fiction books. Reading literature can take readers to new worlds, expose them to fresh concepts and characters, and give them a fresh outlook on life.

In recent years, literary texts like poems, novels, and short stories have been generated and analyzed using natural language processing (NLP) methods. To create new texts that closely resemble the style and substance of the original works, NLP models can learn the patterns and structures of language used in literature. The way we approach writing and storytelling could be fundamentally altered by these frameworks.

The goal of this project is to use NLP to extract short tales from fanfiction archives. Fans of well-known movies, TV shows, books, and video games can create and share fanfiction on websites called fanfiction archives. These stories are set in the same fictional world. Fanfiction is a special kind of writing that blends imagination, fandom, and a passionate attachment to the original work. A rich source of data for NLP projects on narrative development, fanfiction archives offer a variety of stories, characters, locations, and themes.

The objective of this project is to create an NLP model that can produce captivating and intriguing short tales using fanfiction archives. We use the HuggingFace API dataset, which has a substantial amount of fanfiction, to accomplish this objective. Recurrent Neural Networks (RNNs) and Transformers from the HuggingFace packages are two additional kinds of models that we implement. Evaluation will have to be done using manual assessment since most of the acceptable metrics would not work with story generation.

Word clouds will be used to evaluate vocabulary, and train-validate loss monitoring graphs will be used to watch the performance of the models. Data preprocessing, model selection, model training, model evaluation, and result visualization are all included in the project's approach. The HuggingFace API Tutorials, generative model implementations, and Kaggle tutorials were used as sources and guides for the project.

In conclusion, with the right dataset and resources, an NLP model for story generation from fanfiction archives can potentially create intriguing and compelling stories. An exciting area of research involves using fanfiction archives as a source of information for NLP projects on narrative development. This work has potential uses in a variety of industries, including entertainment, digital storytelling, and creative writing.

2 Background/Related Work

Story Generation from Sequence of Independent Short Descriptions, created by Parag Jain, and Priyanka Agarwal, was one of the many efforts in the field of short story generation using NLP. To produce original, creative, and concise language, this article discusses the limitations of the present Natural Language Generation systems. It centers

on the challenge of creating cohesive stories out of fragmented accounts of a scene or event. To address this issue, the article investigates two well-liked text-generation paradigms: Statistical Machine Translation and Deep Learning. On an openly accessible dataset, machine translation and summarization evaluation metrics are used to show the effectiveness of these methods.

Another basic understanding of AI-generated stories was described by Mark Reidl in An Introduction to AI Story Generation. But for a complex understanding, we can take a look at Angela Fan, Mike Lewis, Yann Dauphin- Hierarchical Neural Story Generation. The goal was to develop tools that can create stories from writing prompts. To allow hierarchical story generation, where the model first creates a premise and then transforms it into a coherent passage of text, researchers gathered a dataset of 300K human-written stories and prompts. They incorporated a new mechanism to model long-range context and used a novel type of model fusion to increase the story's relevance to the prompt. By a factor of two to one, human judges favored their stories over a non-hierarchical model, and their method demonstrated notable improvements over strong baselines.

One of the more advanced applications for story-telling included work by Min, Dang - Deep Learning-Based Short Story Generation for an Image Using the Encoder-Decoder Structure. This research introduces the Short Story Captioning (SSCap) AI framework, which creates a short story for an image using an encoder-decoder model and recurrent neural network. In contrast to existing systems that can only create short captions, the framework uses a manually compiled story corpus with two distinct genres (romantic and horror). This could contribute to the creation of a more capable AI story writer and the incorporation of the model into useful applications for generating fresh concepts for stories.

3 Data

Access to a number of datasets, including a fanfiction collection dataset, is available through the Hugging Face API. The fanfiction tales in this dataset were scraped from a variety of online fanfiction repositories, including Fanfiction.net and Archive of Our Own. (AO3). Stories from numerous fandoms, including well-known series like Harry Potter, The Lord of the Rings, and Game of Thrones, are included in the fanfiction collection dataset. The fan-written tales span a range of categories, including sci-fi, romance, and action. Each story's title, author, synopsis, and text are all included in the dataset along with other relevant details. It also contains details on the fandom and the story's main protagonists.

Natural language processing (NLP) activities like text generation, text classification, and sentiment analysis can all be performed using the fanfiction archive collection. It can also be used to develop machine learning models for jobs like text classification and language modeling. Developers can simply incorporate the fanfiction archive dataset into their NLP applications by using the Hugging Face API, which offers a convenient interface for accessing the dataset.

The dataset that is being used in this project is a subset of the fanfiction dataset. It includes fan fiction stories written for the real time stories whose title starts with Z i.e. Z nation, Zathura, Zenda, Zero Wing etc. The dataset consists of 3 features or columns namely story, title, category. The ‘story’ feature is the actual short story written by fans. The ‘title’ feature is the title of the fan-made story mentioning the real time story and author’s name. The ‘category’ feature is the real time story on which the fan story was made. Dataset consists of a total of 943 records. There are totally 43 categories of stories on which the fan stories were made. The distribution of the stories according to the categories are projected and displayed in the graph[1]. Most of the stories made are based on ‘Zoey 101’ category and the least are made on ‘Zinda’.

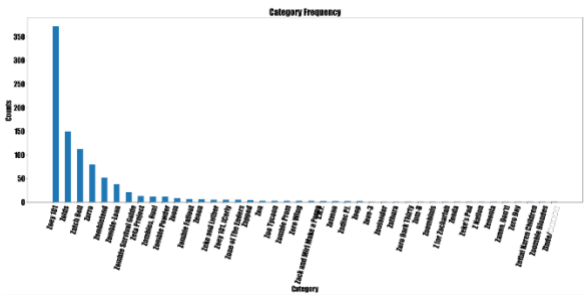

Figure 1: Describing the count of stories per categories.

Word Clouds: The main purpose of finding the word clouds in general is to relate the kind of words and their frequencies that occur per document. For our dataset, we observe how versatile the words are and by extension how many different types of stories can generated using this data. In figures 2 and 3 mentioned below, we can see 2 different categories and their representations of genres through words. Figure 2

showing Z Nation talk about a story involving Romeo and Juliet with the theme of life, love, and death since those words are prominent. Similarly, story of Z for Zachariah in Figure 3 mentions the use of words like God, church and parents evidently portraying its religious inclination. Word clouds could therefore help us visualize themes, enhance situational description, and analyze character traits i.e., what words associate a character and their personalities.

Figure 2 and 3: Describing word clouds associated with 2 different categories of stories – Z Nation associated with words like Romeo, Juliet and love and Z for Zachariah relating to God, church, and parents.

4 Methods

In this project, we are using two methods for generating short stories:

Implementing our own RNN(LSTM) for text generation.

Following steps can be used to generate text using the RNN approach:

Data preprocessing - Data preprocessing is a crucial step in the data analysis process. It involves cleaning and transforming raw data to make it suitable for further analysis. One common preprocessing technique to the symbols used in writing to separate sentences, clauses, and phrases. Examples of punctuation marks include commas, periods, question marks, and exclamation marks. Removing punctuation involves deleting these symbols from the text data. Lowercasing, on the other hand, involves converting all uppercase characters in the text to lowercase. This is done to ensure that the analysis of the data is not affected by the case of the letters used in the text is to remove punctuation and lowercase the words in the text data.

Generating Sequence of N-gram Tokens - For story generation, we need to predict the next word or token in a sequence of words or tokens. The next word will depend on what has been depicted in the story so far. To do this, the first step is to preprocess the text data by tokenizing it. Tokenization is the process of breaking down a text corpus into individual tokens, such as words or terms. Python's Keras library provides a built-in tokenizer model that can be used to extract tokens and their corresponding indices from the corpus. Once the tokenizer has been applied, each text document in the dataset is converted into a sequence of tokens. By converting the text into a sequence of tokens, we can then train a language model to learn the patterns and relationships between the words in the sequence and use this knowledge to make predictions about what the next word or token in the sequence should be used in the text.

Padding the Sequences and labeling the data - Now that we have generated a data-set which contains a sequence of tokens, it is possible that different sequences have different lengths. Before starting to train the model, we need to pad the sequences and make their lengths equal. We can use the pad_sequence function of Keras for this purpose. To input this data into a learning model, we need to create predictors and labels. We will create N-grams sequence as predictors and the next word of the N-gram as a label.

For e.g., Line: they are learning data science.

Predictors	Label
they	are
they are	learning
they are learning	data
they are learning data	science

Training the model - We are using LSTMs as the RNN for sentence generation. LSTMs have an additional state called ‘cell state’ through which the network adjusts in the information flow. The advantage of this state is that the model can remember or forget the leanings more selectively. There are total 3 layers in the LSTM model: Input Layer: Takes the sequence of words as input. LSTM Layer: Computes the output using LSTM units. To begin with, 100 units have been added to this layer. This can be configured more precisely based on further experiments. Dropout Layer: A regularization layer which randomly turns-off the activations of some neurons in the LSTM layer. This layer helps in preventing overfitting of the model Output Layer : Computes the probability of the best possible next word as output. To begin with, the model will be trained for over 100 epochs. **Generating the text** - Finally we use the

trained model to predict the next word. These predicted words can be merged to form a sentence and eventually a story. The model's performance can be further improved by experiments such as fine tuning the network architecture or the network parameters. The model's performance would also depend on how much data it has been trained and validated on, which again will change based on experiments.

Implementing Transformers: Natural language processing (NLP) applications include text creation, text classification, and language translation. These applications employ a type of neural network design called a transformers model. These were initially described in the Vaswani et al. 2017 research "Attention is All You Need." The self-attention mechanism, on which transformer models are built, enables the model to concentrate on various elements of the input sequence while producing the output sequence. The conventional NLP recurrent neural network (RNN) and convolutional neural network (CNN) structures, which analyze the input sequence sequentially, are less effective than this technique. The architecture of transformers can be understood through the figure[1]. The input sequence is initially integrated into a vector space in transformer models before being transmitted via a number of encoder and decoder layers. Each layer has many attention heads that may concentrate on various segments of the input stream. The input sequence is encoded by the encoder layers, while the output sequence is produced by the decoder layers. There are many state-of-the-art transformer models such as BERT, GPT-2, and T5 models. For our project, we focus on custom training the GPT-2 transformer model.

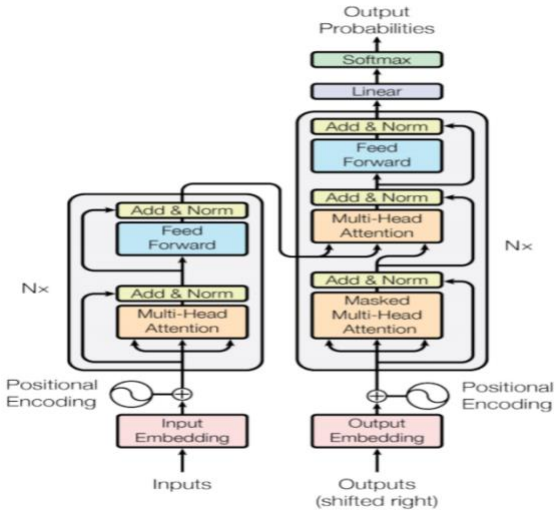

Figure 4

Data preprocessing - The data preprocessing for the GPT-2 transformer is similar to the preprocessing steps of the LSTM.Except, lowering of capitals and handling punctuations were not required. GPT-2 models are immune to them as it was trained on a vast amount of unprocessed text data, which included sentences with mixed capitalization, punctuation, and unpunctuation. All the contractions and special patterns such as repetiveve character, spaces, new line characters were handled respectively in the preprocessing.The tokenization was performed by using the GPT-2 Tokenizer from the HuggingFace API. This method was leveraged as the tokenizer was already trained for English language. The maximum length for the sequences to be processed by the tokenizer was set to 1024. Any sequence more than the mentione length would be truncated.

Training GPT-2 Transformer - For training the GPT-2 model to generate stories, we used the base model from the HuggingFace API. It is just a model trained on English language and do not include any additional fine-tuning or training for specific tasks. We used the GPT-2-small version, which is a smaller size model of 124.5MB. The training is done with 90% of the fan-fiction dataset, the rest 10% is used for validation of the model. The GPT-2 model compatible with TensorFlow framework is used with Adam optimizer to minimize the loss and make the model efficient.

Fine Tuning GPT-2 - For fine tuning GPT-2 model, multiple models were trained with different set of combinations of hyper-parameters. The hyper-parrameters are learning rate, batch size, sequence length, number of epochs . So all models initialized with respective combinations of hyper parameters are trained with training data and evaluated with validation data to find the perplexity of the models.

Text Generation - After fine tuning, the best model with the least perplexity is selected to generate stories. Text generation is done by passing a small prompt, which acts as a seed text. Using this, the model generates the text. The generated stories, randomness can be controlled by setting the temperature parameter of the model.

5 Results

LSTMs: As described in the methods section, the LSTM model was trained on the fan-fiction dataset. The validation percentage was set to 10%. It was observed that the LSTM model performed very well on training data but not so well on the test data. Following graphs show the same(Figure 5 and Figure 6)

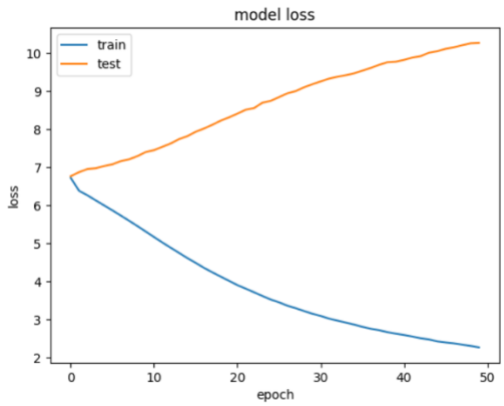

Figure 5

From the above graphs we can see that validation loss increased over the epochs. It may seem like the model was overfitted here. Despite tuning the hyperparameters and changing the LSTM's internal layers, this phenomenon remained the same. Following are a few samples of the LSTM generated stories:

- Just A Question: 2 Aybasbtu Game Script Zero Wing 2 One I Am Not Making My Characters My Doom I Cant Take It Let That Holy Endurance 510 Hound The Early Signs Of This Type Are Lip Splits Rotting Nose Enlarge Eyes Elongated Fingers And Nails Body Becomes Thinner taller Hair Falls Out Smell.
- Alice In Wonderland: Your Remains Rating Zero Wing 2 One I Can Explain Gifts Not A Scientific Man I 1 You Gentlemen Up It Are Like A Lot Oneshot Disclaimer I Need To Be School With My House And Taking Ill The Group Of Justice A Small Metal Guys Had Opened That He

GPT-2:
As mentioned, multiple GPT-2 models were trained with respective combinations of hyper-parameters. The perplexities of these models are shown in Figure 7.

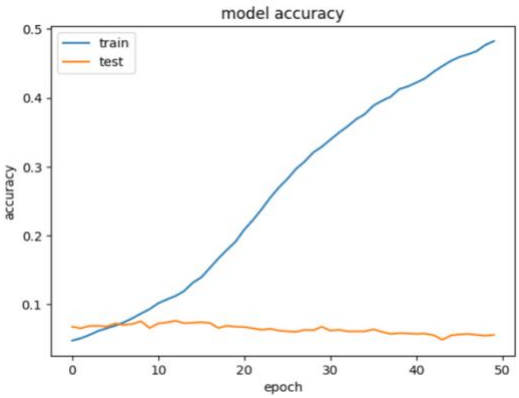

Figure 6

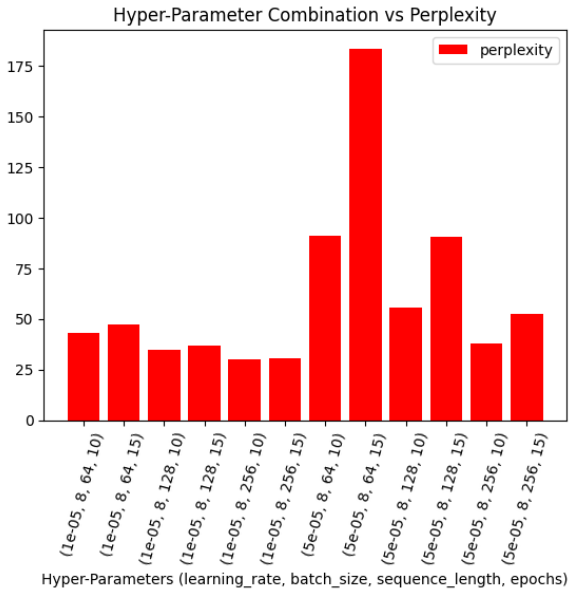

Figure 7

The model with the hyper-parameter combination of learning rate: 1e-05, batch size:8 , sequence length: 256 ,number of epochs:10 was successful efficient with the least perplexity of 29.87916.

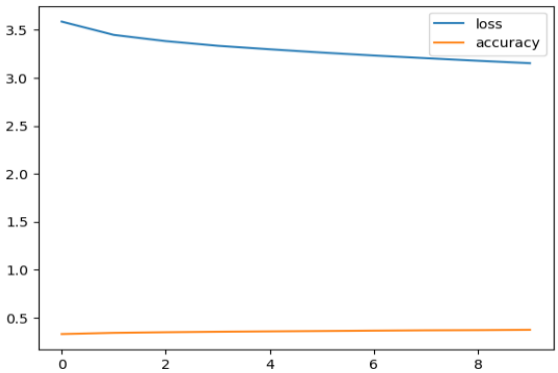

Figure 8

Even though the efficient model's training loss and accuracy were not optimal, as shown in

Figure 8, the model was effective in producing understandable stories that are human readable and had the right context. 4 sample stores were generated for the same prompt by changing the temperature value as shown in the Table 1.

temperature	generated text
1	pain It may be the most beautiful day of my life, but I am afraid it is not going to be for you. I do not know what I will do if I see you again. It is been so long since I last saw you, and I have not seen you in a long time. You are the only one I know that I care about. But I cannot let you go, no matter how hard I try. What I want is you to see me again,
1.5	pain It may be icky but ive never felt ick in my life. I wish I could change that. It is the only thing I can do right now. I do not think that I will ever have the chance to make a difference, even if I have to go back to my old school and do something about it. This is what happened to me when I was about to give up hope of a brighter future and live in a world full of peace and love
2	pain It may be the last day of my life, but then again, I had expected it. I was not here to tell them my real name, not when it would make sense to me; they had decided not to. This time though, all that was important had to be their choice. No one would have thought that such a choice would ever come to pass. And if it had happened, my name would never have been in their hands. It might have ended up in
3	pain It may be The cold blood that seeps out of his veins. But if he did not cry, what should have be in it? For as the wind was blowing to blows, his vision would go dark and his heart would just keep going as his father was struggling with grief. It is this awful, unbearable feeling that comes to him now. No one could stand a more desperate situation in his situation; it is something he can handle without losing his courage and self-confidence.

Table 1

6 Conclusion

LSTMs: The stories generated by the LSTM model were not coherent. Although it generated proper words, the sentences when put together lacked meaning. This could be because LSTM failed to capture the context of the stories.

GPT-2: GPT-2 is able to generate better stories with more coherent and contextually appropriate texts. GPT-2 is better in generating stories compared to LSTM as it is able to capture long-term context and dependencies in text data. Another reason for such good results is the fact that GPT-2 is already a large-scaled pretrained model which is more efficient than traditional language models.

7 References

1. Jain, P., & Agarwal, P. (2018). Story Generation from Sequence of Independent Short Descriptions. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 3980-3989).
2. Reidl, M. (2016). An Introduction to AI Story Generation. In Proceedings of the 2016 International Conference on Computational Creativity (pp. 121-126). ACM.
3. Dang, M. (2018). Deep Learning-Based Short Story Generation for an Image Using the Encoder-Decoder Structure. In Proceedings of the 2018 International Conference on Artificial Intelligence and Computer Science (AICS '18), 145-151
4. Padhi, S., & Bai, K. (2021). Learning Implicit Text Generation via Feature Matching. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 2672-2683).
5. Fan, A., Lewis, M., & Dauphin, Y. N. (2018). Hierarchical neural story generation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 889-898). Association for Computational Linguistics
6. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17) (pp. 6000-6010).
7. Sharma, A. 2021. A Beginner's Guide to Exploratory Data Analysis (EDA) on Text Data (Amazon Case Study). In Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Gold Coast, Australia.)

8. Madhugiri, D. 2021. Exploratory Data Analysis(EDA): Types, Tools, Process. In Proceedings of the ACM India Joint International Conference on Data Science and Management of Data (CoDS-COMAD'21), January 6-8, 2021, Hyderabad, India.
9. Rajapakse, T. (2019). Learning to Write: Language Generation With GPT-2. In Proceedings of the 2019 4th International Conference on Information System and Data Mining (ICISDM 2019) (pp. 221-225).