LSTM Final Report

Colin Yu

Tianze Zhang

Word Count: 1894

Introduction

In the game RimWorld, each character spawns with a backstory that contains a title, a description and skill modifiers. These backstories serve as a critical element in enriching the game's narrative, providing depth to in-game personas. However, the creation of descriptions for backstories is a labor-intensive process if done manually, while being inaccurate if automated using LLMs like ChatGPT.

In response to this challenge, our project seeks to find a compromise between these two, developing a storytelling machine that generates background description for character backstories for RimWorld, tailored to a character's title and skill modifiers, while adhering to the linguistic style of the game's lore and world setting, by fine-tuning a GPT-2 Small, a much smaller model with only 124 million parameters that can run on consumer PCs.

Illustration / Figure

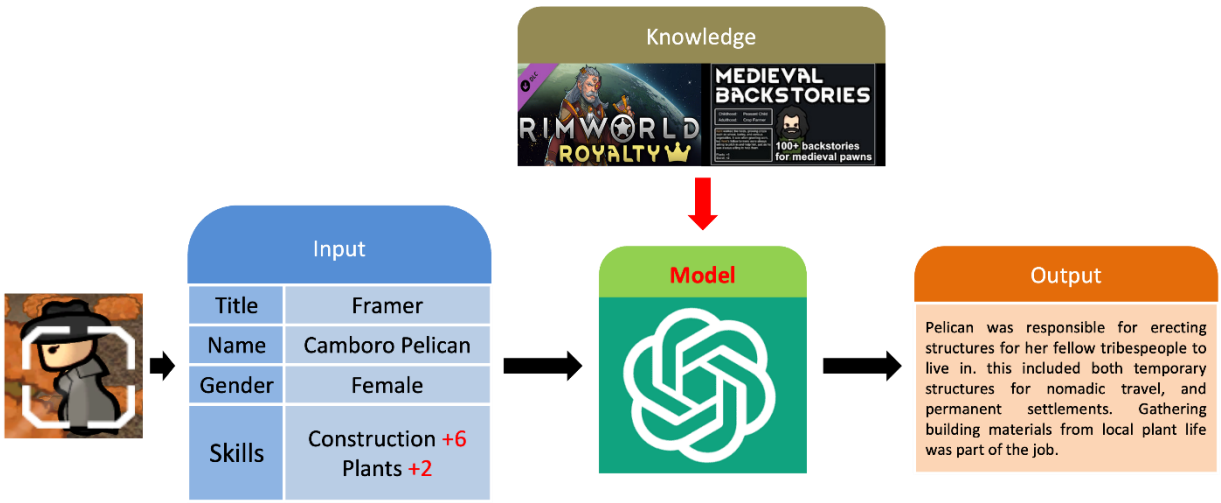


Figure 1. Basic Illustration of LSTM

Figure 1 shows the basic workflow of LSTM. Given a character and its information, the model will output the background description of this specific character. Our main job here is to fine-tune the pretrained GPT model with the backstory dataset that was collected.

Background & Related Work

As of today, there's no other work like this project that takes a title and skill modifiers as input and generates a description for backstories in RimWorld. The closest one is a "Rimworld Backstory Generator" [1] that generates an entire backstory at random, and it doesn't seem to utilize a transformer as the outputs are random selections of predefined lists of words.

However, there do exist projects that use transformers to generate stories. One example of this would be fabled.ai [2], which is a transformer-based story generator that takes a sentence describing the plot and the styles of text as prompt, and generates a story and images based on the given prompt. This approach also had the problem of generating stories that have elements that don't align with the lore of RimWorld.

Data and Data Processing

The backstories are collected from the files of the base game, its downloadable contents (DLCs), and several mods of RimWorld. There are 2073 different backstories in total. All the backstories are stored in XML files. See Appendix A for a screenshot of the XML files. Only 3 parts in each backstory are used for the purpose of this project, and these are:

• Title: The occupation of the character.

• Description: This is what we are trying to make the model generate.

• Skill Modifiers: These are the additional attributes of the character that will alter the generated description based on its value.

Since XML files can't be used directly to fine-tune a GPT-2 model, the information of interest was extracted and stored into a pandas dataframe. Data cleaning was also performed on the description strings by replacing strings like "{PAWN_gender ? he : she}" by "[PAWN_pronoun]", since both means the same thing – "he" or "she" depending on the gender of the character using this backstory. This is to reduce the number of domain-specific special tokens the model needs to learn.

During training, the dataframe is split into a 90/10 train test split using sklearn. A fixed template is used to rephrase each row of data into a training example, which is of the structure: "This is the

story of [PAWN_nameDef], a ”+ Title +“: Descriptions”. One example of a training example will be:

“This is the story of [PAWN_nameDef], a sewer kid: [PAWN_nameDef] grew up in the sludge-smeared sewers of a polluted industrial world.\n\n[PAWN_pronoun] befriended the strange insects in the darkness, and learned to love them”.

Architecture and Software

Our prototype model is a fine-tuned GPT-2 Small model with an LM head. Figure 2 shows the architecture with two branches: the fine-tuning process at the top, and the generation process at the bottom. During fine-tuning, the title, skill modifiers and description will be compiled into a full sentence using the template. Training will stop when the test loss starts to go up in 3 out of the last 5 epochs.

When the model is deployed for actual usage, the input sentence including first part of the template containing title and skill modifier will be fed into the model, and the model will generate a description based on characters title and their different skill modifiers.

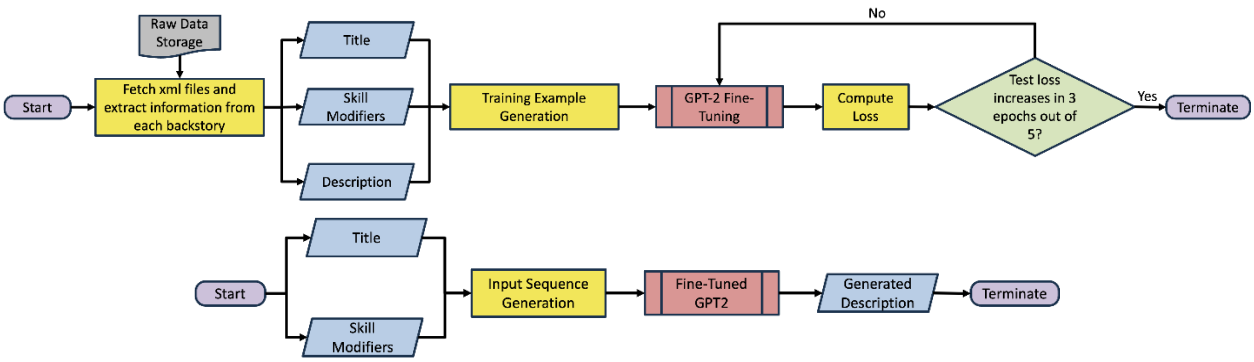


Figure 2. Architecture of LSTM

Most of the libraries that was used for the training are standard machine learning libraries like Pytorch, Transformers, Pandas, Numpy, Sklearn, Matplotlib and Tqdm. Apart from these only the XML library was used, and its purpose was to extract information from the XML files. Gradio was used to build the UI. The details of the user interface can be found in Appendix B.

Baseline Model or Comparison

We chose to compare our fine-tuned GPT-2 Small model with a pretrained GPT-2 Large model to assess whether our fine-tuned model meets our initial goals. To evaluate our model, we established 5 metrics which are listed in the table below in the quantitative results part. To carry out the evaluation, the titles and skill modifiers from the first 100 backstories in the testing dataset were chosen and compiled, and then fed into both model to generate a total of 200 descriptions.

We then manually labeled whether the generated background descriptions meet our criterions and counted the number. If our fine-tuned model generated a visibly larger amount of descriptions that meets a particular metric than the baseline model, it will indicate success in that metric.

Quantitative Results

Metrics	**Fine-tuned GPT-2**	**GPT-2 Large**
Test loss	1.21	3.02
Is the generated description grammatically correct and coherent?	88	61
Are the special tokens like [PAWN_nameDef] used correctly?	93	80
Is the model correctly generating description for the title given?	94	68
Does the generated description fit the game's lore?	95	37
Does the generated description follow the style of game text?	91	27

This table shows the quantitative comparison between our model and the baseline model. It can be seen that our model has greatly improved performance in terms of linguistic style and conformation to the game's lore, and there are also reasonable improvements in terms of the learning of special tokens and generating descriptions that fit the title. However, the grammar quality, although being better than the baseline model, still has space for improvement.

Qualitative Results

Since this project is about text generation, it is decided that it is best to show our analysis of the quality of this project accompanied by examples. Below are two sets of comparisons, with bold green text indicating good performance and bold red text indicating failure.

We will first use the fine-tuned model to generate a description for the title "medieval lord" with skill modifiers "social +7, melee +5, shooting +5", found in our dataset. Here's the output of our model:

[PAWN_nameDef] inherited all the family's wealth when [PAWN_pronoun] was a ***lord in a medieval kingdom****. [PAWN_possessive]* ***social skills improved*** *after a succession of years of rule, and eventually* ***[PAWN_perspective]*** *could be crowned lord and emperor.*

For reference, here's the description for the same title and skill modifiers in our dataset, in the file raw_data/story/basegame/Offworld_Medieval_Adult.xml:

[PAWN_nameDef] was a ***lord on a preindustrial planet****. [PAWN_pronoun]* ***went to parties, managed the underlings****, and even* ***learned some swordplay****.\n\n[PAWN_possessive] soft hands did not hold a work tool during that entire time. [PAWN_pronoun] considers manual labor to be beneath [PAWN_objective].*

The description generated by the fine-tuned model fits the title, and it did talk about the social skill that was given but neglected the melee and shooting skills. This is no big deal though, since the original description didn't talk about the shooting skill modifier either. Aside from this, the only flaw in the generated description is the misused [PAWN_perspective] special token, which did not exist in the dataset.

To investigate whether our model is indeed able to generate different descriptions when the input skill modifiers differ, we will use the model to generate 2 different descriptions for the title "mechanoid operator" – which isn't in the training data – with different skill modifiers.

The description generated with skills "intellectual -3, crafting +4, shooting +2" is:

*[PAWN_nameDef] made robots out of mechanoids, but [PAWN_pronoun] was **adept at the art of manipulation**. [PAWN_pronoun] often used [PAWN_possessive] hands to get into tight systems, and **occasionally would remove a piece of code by bending it or smashing it**.*

The description generated with skills “intellectual +4, crafting -2, shooting +2” is:

*[PAWN_nameDef] grew up in an industrial complex, working on industrial robotics.\n\n[PAWN_pronoun] **learned plenty about robotics**, and the military’s industrial mechanoid army, but **was never particularly good at handling simple machines**.*

[PAWN_pronoun] learned a lot from one mechanoid and a few from another.

From this example it is clear that the model can indeed generate descriptions that reflect the skill modifiers given. For negative intellectual skills, it says that the character occasionally would remove piece of code by mistake, and for positive intellectual skills it says that the character learned a lot about robotics. Similarly, it also talks about how a character with positive crafting skills is “adept at the art of manipulation”, and one with negative crafting skills was never good at handling machines. The shooting skills however, for some reason, got ignored. There is also a factual error in the first description where it talked about physically breaking a piece of code.

Discussion and Learnings

Overall, it should be safe to say that this model has achieved our initial goals. Specifically, when provided with a character and its corresponding skill values, the model generates a comprehensive background description for the character. The descriptions produced by our model surpass those of our baseline model across various metrics and criterion we made, demonstrating superior logical and creative elements.

On the other hand, the handling of skill modifiers was reasonable but not perfect. Sometimes the model simply ignores the skill modifiers given. We concluded that this is due to limitations brought by the dataset since a large portion of the backstories in the dataset have skill modifiers that is unrelated to their corresponding description.

During the development of our model, we also tried adding the special tokens to the tokenizer's vocabulary, but this turns out to be a bad idea as it gives worse results. Later we discovered that this is most likely due to us not setting the model to train the embeddings of these special tokens, so this could be a point for future improvement.

Finally, the length and intricacy of the generated descriptions can also be improved. Currently we have no control over the length of the generated story. The only thing we can do is to set a limit for the maximum number of tokens and hope the generation doesn't end abruptly at the end.

Individual Contributions

Task	Done by
Collecting raw data	Colin
Preprocessing the dataset which turns raw xml files into pandas dataframe	Tianze
Splitting the sentence data and completing the tokenization process	Tianze
Writing the dataset class to load the data	Colin
Writing the collate function for dataset padding	Tianze
Writing the main training loop	Colin
Writing the description generation process of model	Colin
Visualizing the training process for better debugging	Colin
Building baseline model for comparison	Tianze
Writing the Gradio implementation of the user-facing side of the project	Tianze
Conducting the evaluation of the outputs and the comparison of the two models	Both
Writing final report	Both

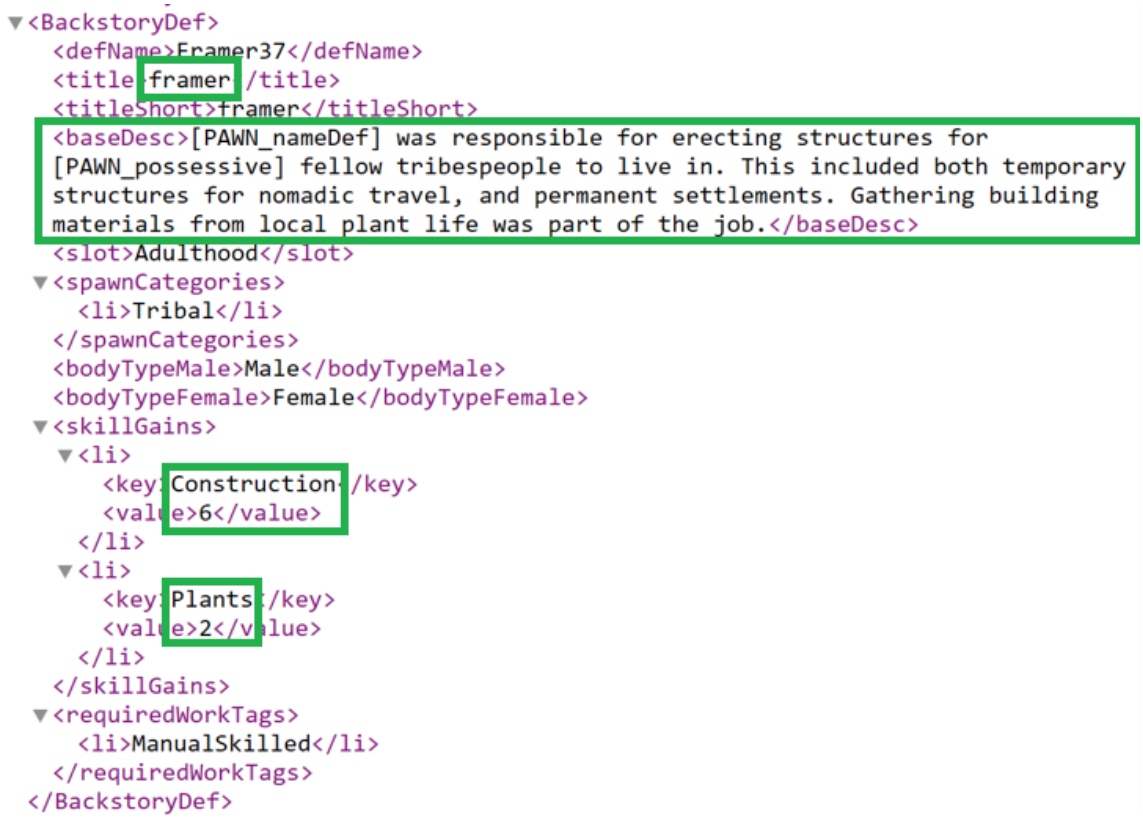
References

[1] “Perchance,” Rimworld backstory generator, https://perchance.org/rimworld-backstory.

[2] Fabled.ai, https://domore.ai/tools/fabled.ai.

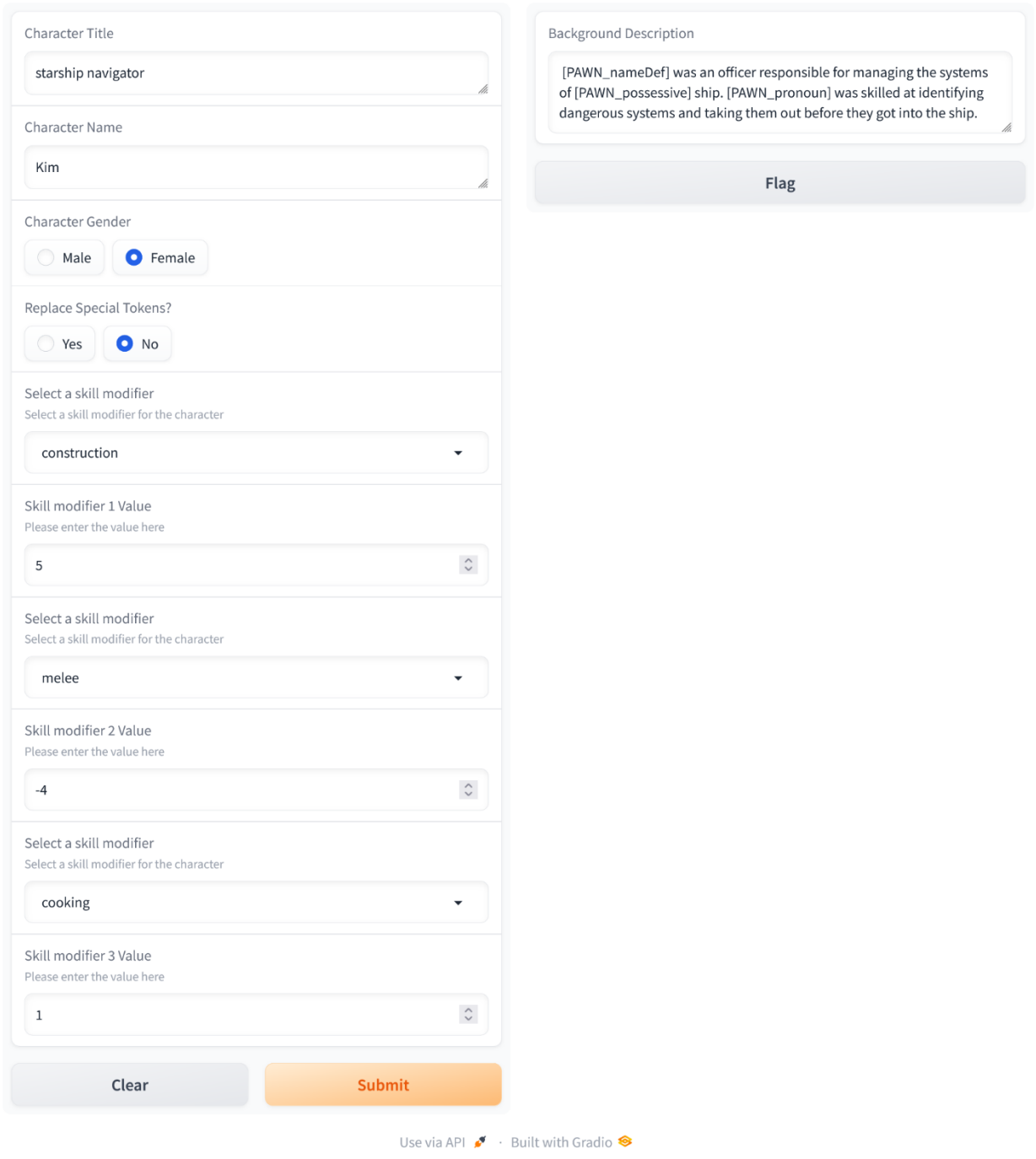
Appendix A

This is a screenshot of a backstory from an XML file. The relevant information is surrounded by green boxes.



Appendix B

Below is the layout of the user interface. The user will input the character title and three skill modifiers ranging from -9 to 9. Optionally the user can also provide the name and gender of the character are replace the special tokens accordingly.



Permissions

Colin Yu:

Permission to post video: Yes

Permission to post final report: Yes

Permission to post source code: Yes

Tianze Zhang:

Permission to post video: Yes

Permission to post final report: Yes

Permission to post source code: Yes