Research Story Generation

Created By	
Stakeholders	
Status	
Type	Research
Created	@July 21, 2022 4:34 PM
Last Edited Time	@July 22, 2022 10:32 AM
Last Edited By	
Property	

In a GPT model, each word is a token and will be referred as such in this document.

GPT Neo - PoC

After making the State of the Art, we realized all organizations working on similar projects used language transformers to generate text.

🌟 State of the Art - AI Story Generator

We therefore tried to implement an open source model. We used GPT-Neo trained with The Pile, a huge 825 GiB dataset.

> The Pile
>
> The Pile is a 825 GiB diverse, open source language modelling data set that consists of 22 smaller, high-quality datasets combined together. The Pile is hosted by the Eye. Have a model that uses or evaluates on the Pile? Let us know!
>
> https://pile.eleuther.ai/

Using short custom made prompts, we ran 125M, 1.3B, 2.7B models. Without surprise. the 2.7B model generated the most coherent stories, but they tend to become less and less coherent on larger generation.

Hardware

We used both our own hardware and google colab.

Our own hardware has significant storage space but low computation capabilities, making text generation slow and excessively slow without a gpu. Training an autoregressive language model on our hardware is not an option.

The advantage of Google colab is that it uses better hardware that what is at our disposal.

But it has limited storage capacity using our google environment, meaning we have to download the model each time we use it. In addition to that, Google colab times out pretty easily, making us start over.

For PoCs and training purposes, google colab (or Kaggle) will be mandatory. If this project went into production, it would need dedicated gpus, no user hardware would be enough to use a gpt based product.

Tokens limitation

When generating a long story, the number of tokens quickly becomes too much for the model to handle. We need another way to generate longer coherent stories. We need the model to ignore a large part of the generated tokens.

Increasing the number of parameters is not a solution that can be repeated endlessly, we quickly reach a hardware limitation.

We then looked for methods used by AI Dungeon and Kobold AI to increase coherence, speed and reduce hardware requirements.

Tokens storage

Kobold AI stores the initial prompt, World building and Author’s notes. World building and Author’s note are formatted differently, but they each store tokens.

https://github.com/KoboldAI/KoboldAI-Client

On every generation, Kobold AI use the tokens from the initial prompt, World building and Author’s notes and the tail of the last generation.

That way, the story stays coherent by following the previous action while still keeping in mind the important information of the story (genre, main characters, relationships, locations).

This solution can be used with or without user inputs, but the story generated stays more coherent when railroaded by the user inputs.

The user can add an input after each generation to lead the story. It’s what AI Dungeon and Kobold AI use for their “Adventure” mode where the story is told at the second person (”You”) and the user write their next action.

It can also be used without user inputs past the initial prompt. The model will generate the story bit by bit while keeping the main information in mind and staying coherent with what it just generated. It can still derail as it will not remember most of what it has generated previously.

In the user input mode, the user is asked to store the world building and author’s note information themselves.

We need to find a way to parse the generated text to extract important tokens about characters, location, events. It will be mandatory if we want to generate an entire story from a single user input.