GPT-2 Based Story Teller

CTEC3451 Development Project

De Montfort University

Artem Bobrov (P2547788)

May 5 2022

Contents

Abstract

The project described in this report is a story-telling application that uses the GPT-2 model to generate text. The model is trained on multiple datasets and is able to generate text from a variety of topics in the backend. The frontend is a web application that allows users to endlessly generate stories accompanied by thematic pictures. The point of the project is to prove the concept that neural networks, if trained properly, are able to generate data that is very similar to human language.

Introduction

Neural networks weren't so popular in the early days of computing, but they are now a common tool for machine learning. First neural network was invented by Frank Rosenblatt in the late 1950s (Rosenblatt 1958). Perceptron has only one layer of neurons, but more advance neural networks can have multiple layers. Before Transformer neural networks there were many different types of neural networks (e.g. Convolutional, Recurrent, LSTM, Feed Forward, etc).

Transformers were introduced in 2017 by Google (Vaswani et al. 2017) and are a new type of neural network. It uses self-attention mechanism to focus on the most relevant information in the input. It weights the significance of each part of the input differently.

This report focuses on the GPT-2 model, which is a transformer type neural network developed by OpenAI in 2019 (Radford et al. 2019). Despite the fact that GPT-3 is a newer model, it's predecessor is still a popular choice for text generation.

An application will be developed that utilises the features of GPT-2 and wraps it up in a user-friendly web interface that allows users to read freshly generated stories accompanied by thematic pictures. The development process will be examined and the results will be presented and discussed in depth. The usage of the key elements and techniques used both in backend and frontend will be explained. Moreover the project development and research cycle will be reflected on and other approaches that possibly could have been used to achieve different or similar results will be presented.

Objectives and Goals

Initial approach

It is important to state that the original plan for the project was to develop a similar application that generates pictures with attached text to them (known as 'memes') and allows users to watch them using a web interface. It has been planned to use different approach to build a repository of training data using 'Optical Character Recognition (OCR)' (Kay 2007) and 'Pillow' library to attach processed text to the pictures. It has been decided to adjust the original plan and change the topic from memes to stories.

New approach

The new approach doesn't require any massive changes to the original plan and the project will be developed using the same techniques - indetical frontend and almost the same backend. The only difference is that OCR will be removed from the backend because it's not needed for the project anymore. To minimize the deviation from the original plan, there will be an additional section in the web interface that creates 'memes' based on novels and other piece of art.

Reasons

Despite the fact that many of the features used in the initial approach have been already developed - Tesseract OCR, Pillow Image placing, 'Meme' scraper - one major issue made it extremely perplexing to continue the project without changing the approach.

The first puzzling part was hidden in reading the text (using Tesseract OCR) from the scraped images and translate them to text strings. Memes contain text captions that makes it easy for a human to read, but this same process is hard for machines to do so - especially via OCR. There were some images successfully processed by it but it didn't satisfy the requirement because the the success rate was too low. Many solutions were tried, but even pre-processing the images before feeding them to the OCR engine didn't seem to give much improvement. The second complexity was to build a meme repository itself. Using the meme scraper is not the best solution because

the content obtained by it can be anything. Unfiltered, uncensored - it's a huge ethical problem.

General Description

Two general components of the development project are described in two sections. Such apparent separation between these categories is due to the fact that this approach allows a simpler, faster and more effiecient implementation with better user experience, it also improves code readability and maintainability.

The first one is the backend, which is responsible for generating text based on given datasets (stories, novels, other pieces of art, etc). It is implemented in Python and uses numerous libraries to achieve the goal. Multiple steps are involved in generating simple text samples including hours of training.

The second component is the frontend, which is a web application that creates a user-friendly, simple interface to scroll and read stories. It is implemented in JavaScript using many libraries, as well as the Python part. Such approach to frontend was chosen because web development is flexible and it's easy to adapt to different platforms. Also it's possible to extend it to a web service that doesn't require downloading the application to user's computer.

The goal is create a system with pre-trained model, which the user can use to scroll through the stories and react to each story leaving a positive or a negative feedback - using the 'Like' and 'Dislike' buttons. Afterwards, these reactions will be stored in a database to be used by the developer to understand which stories should be corrected and improved. The feedback system is designed to create a reflection on the pre-trained model as a whole and not on separate elements (stories). Given the feedback, the developer can decide to improve the model or not.

Major Components

Backend

GPT-2 and the library

GPT-2 is the version of the GPT family used in the project. To operate with the network it was chosen to use a library called 'GPT-2-simple' (Woolf 2021) since it features an easy approach to control and train the model. The library is written in Python and offers a documentation which was used in order to work with the neural network. GPT-2-simple offers two types of models - one with 124 million parameters and one with 324 million parameters. Acknowledging the computational limits of the processing unit used - graphical (GPU), the first model with 124 million parameters was used. It is possible to use the second model but it requires a GPU with significantly more processing power and memory.

It is possible to use techniques that allow to save memory and train bigger models using memory saving gradients - the main memory loss in the computation is due to calculating the gradient of the loss by backpropagation. At the cost of one additional forward pass, the memory consumption can be reduced significantly (Chen et al. 2016). However Tensorflow framework (version 2), which is used by GPT-2-simple, is not optimized for such tasks.

Despite providing simpler control functions, this library also provides a lot of tweaking options. For example, some of the hyperparameters can be changed to improve the model performance during testing. Learning rate, batch size, number of training steps, number of accumulated gradients and optimizer can be changed. Number of layers and other more complex parameters can also be changed but in order to do that, the parameters must be examined and adjusted in Tensorflow code files.

The library also provides a way to load the model from a checkpoint file. This is useful when the model is trained in multiple steps and it's needed to continue training from the last checkpoint without losing progress.

Another important feature for testing and optimising the output is the possibility to use different optimisers. 'Adam' and 'SGD' are available optimisers. The first one is the most popular and is used in the project because the learning time is relatively short and the memory requirements are little (Musstafa 2022). These two aspects are crucial and thereforce the choice fell on 'Adam'.

Datasets

Several datasets are used in the project. It has been chosen to use three datasets to work with. The first one is the novel 'Alice in Wonderland' (Johnsen 2013). This version was taken from GitHub Gist. It's more adapted for machine learning than the original version since it's shorter and therefore contains less tokens. For the reason that it was the first dataset to train the network on, it has been decided to go with the shortened version. The text file with the novel poses a certain inconvenience because it contains a lot of text decoration which can create noise in the model. But also it can have a positive effect on the user experience because it can be perceived as a unique style of the author.

The second dataset used is the novel 'Catcher in the Rye' (Valenzuela 2017). The version found on GitHub is slightly incomplete in the beginning and therefore was filled with the original text from the book. Also it was slightly formated for the sake of the correctness of the output of the neural network. Chapter enumeration and the title page was removed from the text file.

Pillow

Frontend

Web Application appearence

The application appearance is based on Bootstrap 5 and Electron. Bootstrap is a CSS framework that is used to create a responsive and mobile-first design. The main navigation bar that is positioned at the top of the page and can be seen at all times since its fixed is designed using Bootstrap classes.

Displaying the content

User feedback

Database

Linking the frontend and backend

Bibliography

Chen, T., Xu, B., Zhang, C. & Guestrin, C. (2016), 'Training deep nets with sublinear memory cost', *arXiv:1604.06174 [cs]* .
URL: *https://arxiv.org/abs/1604.06174*

Johnsen, P. (2013), 'Alice in wonderland'.
URL: *https://gist.github.com/phillipj/4944029*

Kay, A. (2007), 'Tesseract: an open-source optical character recognition engine'.
Accessed Jan 8 2022 [Online].

Musstafa (2022), 'Optimizers in deep learning'.
URL: *https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0*

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. (2019), 'Language models are unsupervised multitask learners'.
Accessed Jan 6 2022 [Online].

Rosenblatt, F. (1958), 'The perceptron: A probabilistic model for information storage and organization in the brain.', *Psychological Review* **65**, 386–408.

Valenzuela, C. (2017), 'the-catcher-rye'.
URL: *https://github.com/cvalenzuela/rwet/blob/master/final/source_text/ny/the-catcher-rye.txt*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. & Polosukhin, I. (2017), 'Attention is all you need'.
Accessed Jan 7 2022 [Online].

Woolf, M. (2021), 'gpt-2-simple'.
URL: *https://github.com/minimaxir/gpt-2-simple*

Appendices

Appendix A

Appendix B