TV Script Generation using Recurrent Neural Networks (RNN)

Bhuvan Kapoor [23MDT1003]

Abstract

This report presents the development and implementation of a Recurrent Neural Network (RNN) model for generating synthetic TV scripts. The model is trained on a dataset of TV show dialogues and is capable of producing realistic text that mimics the style and flow of authentic TV scripts.

1 Introduction

The generation of synthetic text has become a prominent application of neural networks, particularly in the field of Natural Language Processing (NLP). This project focuses on generating TV script-like text using a Recurrent Neural Network (RNN). By training on a dataset of TV show scripts, the model learns language patterns, sentence structure, and stylistic choices unique to TV dialogue. The application provides an interactive platform, enabling users to input text prompts and generate responses similar to those in TV shows.

2 Project Objectives

The primary objectives of this project are as follows:

- Develop a text generation model based on RNN architecture to generate coherent TV script segments.
- Enable prompt-based text generation that mirrors the style of TV show dialogues.
- Deploy the model in a user-friendly web application using Streamlit, allowing users to interact with the model in real time.

3 Dataset

3.1 Data Source

The dataset consists of script dialogues from popular TV shows. Each line represents a character's dialogue, and the dataset includes metadata such as character names, scene context, and episode information. This dataset provides diverse linguistic structures and conversational patterns, which aid in training a robust text generation model.

3.2 Data Preprocessing

The preprocessing pipeline involved the following steps:

- **Tokenization:** Converting text into individual tokens for sequential processing.
- **Normalization:** Lowercasing, removing punctuation, and cleaning up irregular spacing.
- **Sequence Generation:** Dividing the text into sequences of fixed lengths to be used as input-output pairs in training.
- **Padding:** Ensuring each input sequence has a uniform length for batch processing.

Data preprocessing was essential to transform raw script text into structured input that the RNN could process effectively.

4 Model Architecture

4.1 Recurrent Neural Network (RNN) Structure

The model was built using a multi-layered RNN structure with Long Short-Term Memory (LSTM) units to capture long-term dependencies in text. The architecture includes:

- **Embedding Layer:** Converts words into dense vectors of fixed size, enabling the model to process textual data numerically.
- **LSTM Layers:** Two LSTM layers with dropout for regularization, improving model generalization.
- **Dense Layer:** A fully connected layer with a softmax activation to generate a probability distribution over possible next words.

4.2 Model Hyperparameters

The model was trained using the following hyperparameters:

- Learning Rate: 0.001
- Batch Size: 64
- Epochs: 20
- Dropout: 0.2 (for regularization in the LSTM layers)

5 Training Process

5.1 Training Strategy

The model was trained to minimize the categorical cross-entropy loss, with the Adam optimizer managing the learning rate adaptively. Early stopping was implemented to prevent overfitting.

5.2 Evaluation Metrics

The primary metric for model evaluation was perplexity, measuring the model's ability to predict a sequence of text. Lower perplexity indicates better predictive performance. Validation loss was also monitored as an indicator of model generalization.

6 Results and Analysis

The model demonstrated the ability to generate text sequences that are syntactically and contextually relevant to the input prompt. Below are some generated text examples after training:

- **Example 1:** *"Well, I guess we're just going to have to wait and see what happens. Right, Jerry?"*
- **Example 2:** *"Look, it's not about the money, it's about the principle!"*

6.1 Performance Metrics

The final perplexity on the validation set was observed to be approximately 30.5, indicating that the model has learned to produce coherent and contextually appropriate text.

6.2 Error Analysis

Generated text occasionally contained repetitive phrases or incoherent sequences. This can be attributed to limited diversity in training data or insufficient training time. Future improvements could focus on increasing the data size and further tuning model hyperparameters.

7 Conclusion

This project successfully demonstrates the application of RNNs for text generation, specifically in the domain of TV scripts. The model is able to generate coherent dialogue that resembles real TV scripts, although some improvements are possible in terms of diversity and coherence. Potential future work includes:

- Expanding the dataset to include a broader range of TV show scripts.
- Experimenting with Transformer models for potentially better performance.
- Integrating additional customization options, such as adjusting the script's tone or character traits.