Recurrent Rhapsody: A Prompt-Driven Song Generation Pipeline

Harshita Chadha
harshitachadha@gwu.edu
Computer Science Department
School of Engineering and Applied Science

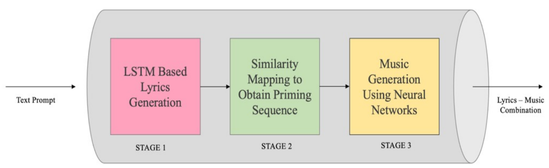
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Abstract

The project titled "Recurrent Rhapsody: A Prompt-Driven Song Generation Pipeline" is a Recurrent Neural Network (RNN) based three-stage architecture for music generation. The project is aimed to produce an enhanced intelligence system by combining several machine learning algorithms such as Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and sentence-BERT embeddings. At each of the three stages, models performing specific tasks are trained on separate but interlinked datasets and the results produced are compiled and presented as a coherent desired output. The experiments carried out during the project development culminate in a sequential pipeline that, given a textual prompt, generates a complimentary song- lyrics combination.

Introduction

Recurrent Rhapsody is a three-model pipeline that can, given a text prompt, output a unique lyrics-song combination. A central assumption that is followed through in the project is that in a generated song, the tone of the lyrics matches that of the backing track. In order to achieve the desired results, three separate models were trained and then interlinked in such a way so that the output of one of the model feeds the input of the next. A diagrammatic representation of the architecture of the pipeline is shown below –



The first part, an LSTM model generates lyrics of a song, given a prompt. This song is then fed into the second stage, sentence-BERT embedding and cosine similarity-based algorithm, where a similarity mapping is done to obtain the song with the most similar lyrics to the one generated. The MIDI track for this resultant song is then fed into a neural network model as the priming sequence to generate a new backing track. The lyrics along with this track are then output as the results of the pipeline ready to be combined.

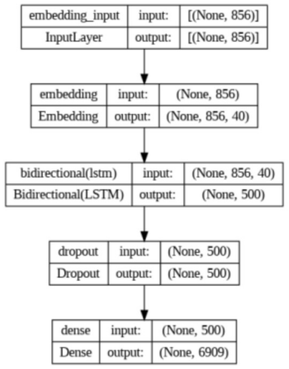
Methodology

The first task performed during the development of the project was the identification of the data sources that will be used to train the model. For the first component, which at its core is a text generation model, a song-lyrics dataset had to be utilized. The data was sourced from a Kaggle repository that contained songs scraped from 79 genres from a Brazilian music website called Vagalume. A snippet of this dataset is shown below –

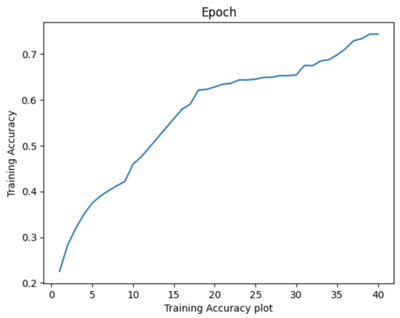
	ALink	SName	SLink	Lyric	language
0	/ivete-sangalo/	Arerê	/ivete-sangalo/arere.html	Tudo o que eu quero nessa vida,\nToda vida, é\...	pt
1	/ivete-sangalo/	Se Eu Não Te Amasse Tanto Assim	/ivete-sangalo/se-eu-nao-te-amasse-tanto-assim...	Meu coração\nSem direção\nVoando só por voar\n...	pt
2	/ivete-sangalo/	Céu da Boca	/ivete-sangalo/chupa-toda.html	É de babaixá?\nÉ de balacubaca?\nÉ de babaixá!...	pt
3	/ivete-sangalo/	Quando A Chuva Passar	/ivete-sangalo/quando-a-chuva-passar.html	Quando a chuva passar\n\nPra quê falar\nSe voc...	pt
4	/ivete-sangalo/	Sorte Grande	/ivete-sangalo/sorte-grande.html	A minha sorte grande foi você cair do céu\nMin...	pt
5	/ivete-sangalo/	A Lua Q Eu T Dei	/ivete-sangalo/a-lua-q-eu-t-dei.html	Posso te falar dos sonhos, das flores...\nde c...	pt
6	/ivete-sangalo/	Mulheres Não Têm Que Chorar (com Emicida)	/ivete-sangalo/mulheres-nao-tem-que-chorar-com...	Hey, gir\nLevanta da cama\nO que foi que te a...	pt
7	/ivete-sangalo/	Eva / Alô Paixão / Beleza Rara	/ivete-sangalo/eva-alo-paixao-beleza-rara.html	"EVA"\n(Giancarlo Bigazzi/Umberto Tozzi)\n\nMe...	pt
8	/ivete-sangalo/	Flor do Reggae	/ivete-sangalo/flor-do-reggae.html	Um brilho de amor chegou na ilha inteira\nE a ...	pt
9	/ivete-sangalo/	Carro Velho	/ivete-sangalo/carro-velho.html	Cheiro de pneu queimado, carburador furado, co...	pt

Cleaning of this dataset was performed, and the lyrics were tokenized and then converted into n-gram sequences for the text generation task. For the 2nd component of the pipeline, the MIDI tracks to be used in the final stage had to be referenced and the IDs for each of these tracks was the identified. Using these IDs, the Musixmatch API for lyrics extraction, the langdetect library for language detection, and some metadata files found in the repository of the lakh MIDI dataset (the parent dataset of the lakh pianoroll dataset – component 3's data) a new data file was generated, and this new data file served as the basis for component 2 of the pipeline. The data for component 3 came from the lakh pianoroll repository that contains MIDI tracks for multiple different songs. The LPD-5 cleansed version of this dataset was used to train the model in component 3.

As for the models themselves, the component 1 model is a bidirectional LSTM model made up of 4 layers as shown below –



The first layer is the embedding layer that maps the input n-gram sequences into dense vectors that are then fed into the Bidirectional LSTM layer that is made up of 250 units. The third layer is the dropout layer that helps prevent overfitting and maintain generalization. The last layer of the model is the dense layer which has a SoftMax activation function that outputs the probabilities of each word in the vocabulary to be the next word in the sequence. This model was then compiled using the categorical cross entropy loss function and Adam optimizer. The training process was 40 epochs long and the batch size was 32. The model achieved a training accuracy of 74.36% and the accuracy plot for the model is shown below -



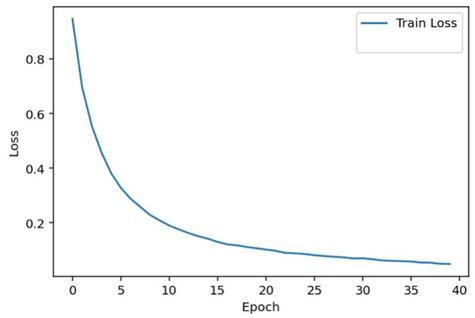
The second component of the pipeline is responsible for the identification of the priming sequence required for backing track generation by the third component's Recurrent Neural Network (RNN). This operation is performed so that music complementary to the lyrics generated by component 1 is produced by the pipeline. After creating the new relevant datafile for this component as described in the preceding paragraphs, we use the sentence transformer library which is built on top of Huggingface's transformers library to load a pretrained BERT based model namely the 'bert-base-nli-mean-tokens', i.e., the Sentence-BERT, to generate embeddings for each of the lyric entries in the dataset. These embeddings were then saved as a pickle file. With these embeddings at hand, given a text, the same pretrained model can be used to generate embeddings for this text. Following this, the cosine similarity between each of the embeddings in our saved file and the new embedding is calculated, and the saved lyric embedding for which this score is the highest is identified as the source of the priming sequence to be fed into the third stage of the pipeline – the backing track generation model.

Given the generated lyrics and a complementary priming sequence in the available MIDI dataset, the next step is to create an RNN model that is capable of taking in this priming sequence and generating a completely new backing track. This was achieved in the third stage of the pipeline. The model used in this component is not a vanilla RNN but rather an advanced version of it called GRU or the Gated Recurrent Unit . The model is made up of four layers in total and the summary is illustrated in the figure below -

```
print(model_comp3)
```

```
GenerationRNN(
  (embedding): Embedding(320, 96)
  (gru): GRU(96, 96, num_layers=2)
  (decoder): Linear(in_features=192, out_features=320, bias=True)
)
```

The first is the embedding layer which takes in the integer list of notes and converts it into a dense vectors. These are then fed into the two layers of the gated recurrent unit and the output of the gated recurrent unit then decoded to produce a probability distribution of over the possible notes.



The loss function used during training is the cross entropy loss and the SoftMax activation function is used during evaluation to pick the note with the highest probability from amongst the produced distribution. A plot of the observed loss over the training epochs is shown above. The model trained for 40 epochs and the training terminated when the observable loss value was about 0.05.

The entire project was created using Python 3.10.11 and its various libraries. The first component which is an LSTM model for text sequence i.e. lyrics prediction was built using tenserflow.keras. The second component used to obtain a complementary priming sequence was built using a pretrained sentence-BERT model from huggingface interfaced using the sentence tokenizer library. The third component which was a Gated Recurrent Unit based model was built using PyTorch.

The first and second components of the model were trained in the Google Collaboratory environment using the NVIDIA Tesla T4 GPU. The third component was trained using Jupyter on the local machine using a 1.1 GHz Quad-Core Intel Core i5 processor and no dedicated GPU.

Results

Once each of the three models was trained and prepared separately, the task was to combine and pipeline these components together sequentially so that the desired output could be produced. In order to perform this task a function was created that references and loads each of the pretrained model and makes sure the appropriate operations are being carried out so that the output of each t-1th component is compatible with the input dimensions of each tth component.

The image above is used to illustrate a sample output that is produced when a text prompt – "I hate my life" is given to the the Recurrent Rhapsody pipeline.

Conclusion

The aim of this project was to develop a pipeline that utilizes the power of neural networks to generate a new song based on a given text prompt. This was achieved by combining several machine learning algorithms, including LSTM, GRU, and Sentence-BERT embeddings. However, the pipeline's current implementation only utilizes a subset of the available data due to training infrastructure limitations, and incorporating more data could lead to even better performance. One way to achieve this would be to add more training examples for the LSTM and similarity mapping algorithm in the first two components. In the third component, integrating tracks for multiple instruments and using VAE-based neural networks for more accurate next note predictions could improve performance.

Although there is room for improvement in the project's output quality, the current level of performance is satisfactory. Overall, this pipeline presents a novel approach to music generation that has exciting real-life applications

References

[1] Dong, Hao-Wen & Hsiao, Wen-Yi & Yang, Li-Chia & Yang, yi-hsuan. (2018). MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment. Proceedings of the AAAI Conference on Artificial Intelligence. 32. 10.1609/aaai.v32i1.11312.

[2] Song lyrics from 79 musical genres. (2022, March 17). Kaggle. https://www.kaggle.com/datasets/neisse/scrapped-lyrics-from-6-genres

[3] Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., & Lamere, P. (2011). THE MILLION SONG DATASET. In International Symposium/Conference on Music Information Retrieval (pp. 591–596). https://doi.org/10.7916/d8nz8j07

[4] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1908.10084