Título: Construção de modelo gerador de letras musicais

Autor: Matheus Freitas Sant'Ana

Link repositório: https://github.com/MatFreitas/nlp-projfinal

1. Introdução

Esse projeto tem como objetivo desenvolver um gerador de letras musicais com base em um dataset do Genius, em que as pessoas podem dar upload de letras musicais, poesias, e derivados de maneira democrática. Dessa forma, esse modelo poderia ser uma ferramenta útil para músicos em seus processos criativos, os ajudando a tornar a criação de letras mais eficiente e simples.

2. Preparação dos Dados

O Dataset possui um grande número de dados, contendo não só a letra de cada música, como outras informações como também a qual gênero pertence. O processo de limpeza dos dados consistiu basicamente de limpar a pontuação das letras e tirar as colunas desnecessárias.

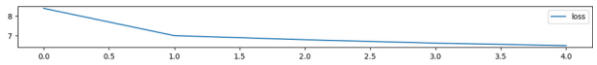
3. Modelos

a. NLTK

Para o primeiro modelo, foi utlilzado um preditor de palavras do NLTK com base em n-gramas. Assim, as frases de cada documento precisavam ser separados em n-gramas, adicionados de um "padding" especial que era feito com símbolos especiais. Depois desse pré-processamento, e treinar o modelo, era possível gerar letras. Como esse modelo não utiliza redes neurais, sua validação foi realizada de maneira empírica, e, de fato, as letras geradas continham uma natureza similar àquelas do dataset (a maioria das músicas era de rap e tinha um vocabulário característico).

b. LSTM

Esse modelo é uma RNN, cujo número total de parâmetros era mais que oito milhões. Devido à natureza do dataset (impossibiltando o uso da função que já cria um tensor dataset a partir dos dados), os dados foram processados de uma maneira alternativa. A tokenização foi feita de maneira similar ao dos n gramas, no entanto foi necessário gerar uma sequência de tokens, processo que replica aquele do text vectorizer. Como o tamanho das sequências ficou muito grande, o que excedia a memória do notebook, utilizou-se apenas parte delas. No final, por não utilizar o vectorizer, não foi possível implementar uma janela de contexto deslizante, resultando com que ele repita as mesmas palavras. Aqui segue o gráfico de loss do modelo:



c. Bloom Model

Por fim, o último modelo, que foi obtido a partir do hugging face, foi o bloom. No caso, mais especificamente falando, utilizou-se o beam search que gera a sequência de palavras (de tamanho n) mais provável. Ele teve o resultado mais satisfatório dentre os modelos.

4. Conclusão e possíveis iterações

No repositório citado no começo desse relatório, existe um arquivo Jupyter em que os 3 modelos podem ser testados, Além disso, também foi criado um mini servidor em Flask para conseguir testar os modelos, no entanto, foi possível apenas subir o modelo NLTK, os outros dois estavam com problemas das versões do keras/tensorflow (as linhas que iriam criar as interações com os modelos estão comentadas no código-fonte).

Como melhorias, pode-se focar no LSTM na questão de implementação da janela deslizante. Além disso, muita das letras possuem um vocabulário de baixo calão, poderia-se filtrar essas palavras.

5. Referências

https://towardsdatascience.com/getting-started-with-bloom-9e3295459b65,
https://www.kaggle.com/code/shivamb/beginners-guide-to-text-generation-using-lstms,

https://www.kaggle.com/code/alvations/n-gram-language-model-with-nltk