Lyrics Generator: Project Report | SI 630

Prathamesh Joshi
University of Michigan
prathuj@umich.edu
April 21, 2023

Abstract

With many attempts in modern day NLP for text generation, this project attempts to create a lyric generator with character based LSTM model. With direct advantages of character based model over word based models and that of LSTM over RNNs, we apply this in Pytorch library. The project is concluded using BLEU as an evaluation metric and followed up with discussions for further improvements.

1 Introduction

I have proposed to work on a lyrics generator which will generate a lyric of specified character length based on a given input string. A lyricist can often have a mind block while considering verses or may even find it difficult to get an idea. The principal reason for this exploration is to enable the lyricist to get a motivation that can assist him in making better verses.

2 Data

The data is obtained from Open Lyrics Database [1]. This data contains lyrics of songs and some translations stored in folder structure. For the training of the model, I have taken 75 files from the data. Each file contains lyrics of one song. The files were combined into one file after appending the file names and rewriting the individual file data into one single file. This data was then used for training purpose. Data prepossessing includes steps like:

- Converting the text to lower case
- Removing punctutaions (.,{}@)
- Replacing all other the special characters like \n \t etc., with spaces

which were done using the Regular Expression[2] python library

[1] https://lyrics.github.io/
[2] https://docs.python.org/3/library/re.html

2.1 Data Selection

From the first impression of the task, we get a primary understanding that this problem is an application of Multi-class classification problem. The lyrics generation task could be achieved by multiple strategies. These strategies also vary according to the choice of the model, representation of data, construction of classes, choice of the tokens, implementation of training algorithms etc. I have articulated my approach of the first implementation of Lyrics generator task in Section [8] (Other things I tried) and explained why the approach was not suitable for robust implementation.

In this implementation, I have attempted to tackle this on the basis of character level Long Short Term Memory (LSTM) model. Multiple character Level LSTMs have been utilised previously but this problem is unique in the case that the output vector is also of the same dimensions of the input vector.

1. A string of contiguous characters of length 250 is chosen randomly from the text as input vector.

2. The next 250 corresponding characters from the text are chosen as output vector

For example: For a character LSTM with input size 6, and a sentence: "A cat ate the rat", a random input could be `['c','a',t',' ','a','t']` and the corresponding output vector would be `['a',t',' ','a','t','e']` converted to tokens.

3 Related work

Diksha Khurana, Aditya Koli, et al., (1) have worked on understanding the current challenges and trends in lyrics generation by applications of natural language processing. Automated generation of song lyrics was studied by S. Pudaruth, S. Amourdon and J. Anseline (2) with the use of

CFGs. Previous research in the field of computational linguistics has focused on lyric generation for specific genres, limited to Recurrent Neural Networks (RNN) or Gated Recurrent Units (GRU).

Venkat Chadalavada (3) has attempted to make a lyrics generator by using both character level and word level RNN. He concluded that character-based models perform better than word-base. Even though word embedding is a very innovative method in NLP, word vectors by itself hardly convey semantics efficiently. Jayashree Domala, Manmohan Dogra, and Anuradha Srinivasaraghavan (4) have attempted to use a bilinear LSTM model for lyric generation and have also attempted to make a comparision between LSTM, Bidirectional LSTM and GRU models. However the model is based on word level LSTM and uses padding sequences for getting a uniform input and output vector. The use of LSTM network to produce lyrics for a specific genre has been attempted by Harrison Gill, Daniel Lee, Nick Marwell (5). They have also attempted to evaluate the generated lyrics via several linguistic metrics and compare these metrics to those of other genres.

4 Methodology

For the application of multi-class classification, the 250 character input and output vectors are converted into a vector representation of 250 vectors each vector representing one-hot encoded vector for all alpha-numeric characters in English taken from the `string`[3] module.

I have fit an LSTM model for the Multi-class classification problem. The number of characters from `string.printable` form `string` module is set as both the `input_size` and `output_size` of the model. The model fits:

1. an Embedding [4] layer over the input vector with `hidden_size = 128`
2. passes it through the LSTM [5] structures with `num_layers = 2`
3. and a fully connected Linear [6] layer

I have implemented this model in Pytorch package which has easy to modify hyper-parameters and compatible neural network structures.

[3] string — Common string operations
[4] nn.Embedding
[5] nn.LSTM
[6] nn.Linear

4.1 Implementation and hyper-parameter tuning

The implementation of the Lyrics generation is done by the following steps:

1. A `get_random_batch` selects a 250 character contiguous vector as the input vector and following 250 character contiguous vector as the output vector
2. A `chat_tensor` converts the input and output vectors to one-hot encoded vectors
3. A `train` loop which iterates over the `LSTMModel` for 10,000 epochs for tuning of the model
4. A `generator` function generated the lyrics based on a `input_string` and `prediction_len` provided. The function also uses `temperature` parameter which determines uniqueness of lyrics generated. I have set the parameter `0.65` as a trade-off between generation of common words and also to prevent generation of non-existing words.
5. The `batch_size` was set to `1`. It was found out that changing the `batch_size` did not affect much of the training process.

5 Evaluation and Results

5.1 Loss

For the training process I have set the `batch_size = 1` The model was trained for 10,000 epochs and optimized using Adam [7] optimizer gave a CrossEntropyLoss [8]. The loss was plotted corresponding to every 500 epochs and lyrics were also generated for every 500 epochs. Some of the instances for generated lyrics and loss are shown in Figure [3].

The Tensorboard plot in Figure [1] shows a decreasing loss which indicates that our model is performing well with the chosen learning rate.

5.2 BLEU Score

BLEU (6) (bilingual evaluation understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Quality is considered to be the

[7] optim.Adam
[8] nn.CrossEntropyLoss

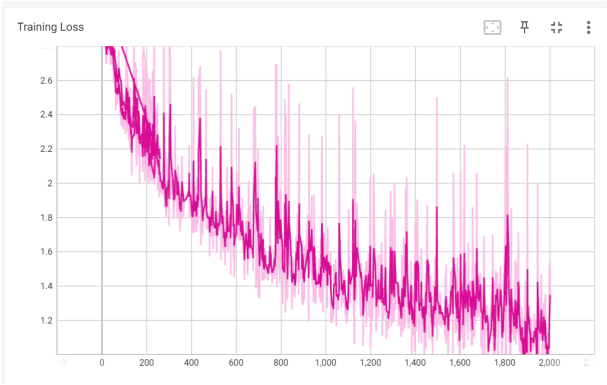

Figure 1: Tensorboard plot of loss v/s epochs

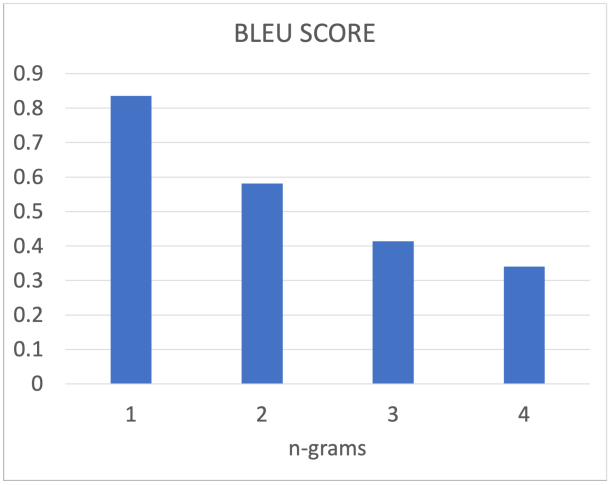

Figure 2: Cumulative BLEU score v/s n-grams

correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" – this is the central idea behind BLEU. The BLEU was computed for 1-gram, 2-gram, 3-gram and 4-gram cumulative cases and the agerage over the training epochs is shown in the Figure [2].

5.3 Baseline

Due to the constraints of time and my limited knowledge in the field, I am not able to display the performance with respect to the baseline. Generally, the baseline for the model is a language model based on n-grams. An n-gram language model predicts the probability of the next word in a sequence based on the previous n-1 words. Computation of Perplexity for model can also be used as a metric of evaluation.

6 Discussion

The advantage of using this strategy is that in comparison to the word based tokens, character tokens have an ability to handle rare words or out-of-vocabulary words. This is because words that are not present in the vocabulary can still be broken down into their constituent characters, which can be used as input features. This is particularly useful for languages with complex and variable morphology or technical domains with specific jargon. Character-level models can capture more fine-grained representations of the text compared to word-level models.

Another advantage is that, LSTMs have been shown to outperform standard RNNs on a wide range of complex tasks, such as text generation, speech recognition, machine translation, and handwriting recognition. One of the primary advantages of LSTM over RNN is its ability to handle long-term dependencies. LSTMs are designed to allow information to flow through the network over longer periods of time, making them particularly effective for tasks such as language modeling, speech recognition, and machine translation, where it is important to maintain context over longer sequences.

One of the setbacks of application of this strategy is that, though we get a good BLEU score, the lyrics generated do not necessary follow flow of a natural language. This is because the BLEU score does not take into account other important factors such as grammar, syntax, meaning, and coherence, which are also essential for generating good quality text. Character-level models may capture less semantic meaning compared to word-level models, as they only capture the structural information of the text, rather than the meaning of individual words. Also character-level models may have difficulty capturing long-term dependencies, particularly for tasks such as language modeling where the model needs to remember previous words or characters in the sequence.

The evaluation of model can be performed in multiple ways. One way is to try feeding in the first half of a song and see if the model can come close to generating the second half (using ROUGE or BLEU). Human evaluation could be even a better evaluation. However, this evaluation did not work well, when I try to reuse the saved trained model for the generation of lyrics. I have included the code snippet for the following structure in Listing [1].

7 Conclusion

This project presents a character-level LSTM (Long Short-Term Memory) model for generating lyrics. The model is trained on a large corpus of lyrics and is capable of generating lyrics that resemble those of the training corpus though it still lacks the coherency and semantic meaning. I have evaluated the model using the BLEU (Bilingual Evaluation Understudy) metric, which measures the similarity between the generated lyrics and the original lyrics. I have also discussed the advantages and limitations of using a character-level model over a word-level model for lyrics generation, as well as potential directions for future research.

The Github link to the Jupyter notebook is at: Lyrics Generator Final.ipynb

8 Other things I tried

As a first step to analyse the task, as seen from many of the existing lyrics generation applications available on the web, I have tried to implement it by using a Logistic regression model. The tokens were chosen as words considering the entire vocabulary of the lyrics data file. As this would increase the vocabulary to a large extent, I set a minimum frequency filter to limit the vocabulary size.

Since the data was in format of multiple lines of a large lyrics data file, each sentence was considered as data for supervised learning. Each consecutive word became a label for its previous word. For example, in the sentence, "A cat ate the rat", for inputs be `['A','cat','ate'],['A','cat','ate','the']` the corresponding output vector would be `['the'],['rat']` respectively converted to tokens. Each input sequence was also padded to make it of uniform length.

The outputs in this case were more lyrical (Figure [4]), however this model lacked the capacity to increase the vocabulary size, and that would negatively impact the learning of the Logistic Regression model, thus leading to generation of repetitive words in the output.

9 What I would have done differently

I would have tried to incorporate semantic information into a word-based LSTM model. One way to incorporate semantic information is to pre-train the LSTM on a large text corpus using an unsupervised learning approach, such as word2vec or GloVe. Determination and evaluation on Baseline is to be worked upon. The cosine similarity between the metric vectors for original and generated songs by can be a good metric to evaluate the model performance.

Listing 1: Code for saving the trained model and using it for generation

```
def save_model(self):
        with open('filepath1', 'wb') as f:
            pickle.dump(self.model, f)
def save_model_state_dict(self):
    torch.save(self.model.state_dict(),
    'filepath2')
def generate(self, temperature = 0.65):
        with open('filepath1', 'rb') as f:
            loaded_model = pickle.load(f)
        modelSD = loaded_model
        modelSD.load_state_dict(torch.load
        ('filepath2'))
```

References

[1] Khurana, D., Koli, A., Khatter, K. et al. Natural language processing: state of the art, current trends and challenges. Multimed Tools Appl 82, 3713–3744 (2023) https://doi.org/10.1007/s11042-022-13428-4

[2] S. Pudaruth, S. Amourdon and J. Anseline, Automated generation of song lyrics using CFGs, 2014 Seventh International Conference on Contemporary Computing (IC3), Noida, India, 2014, pp. 613-616, https://doi.org/10.1109/IC3.2014.6897243

[3] Venkat Chadalavada, Lyrics generator using LSTM, https://medium.com/@venkat.chadalavada/lyrics-generator-using-lstm-e174bfc6c2f

[4] Jayashree Domala, Manmohan Dogra, and Anuradha Srinivasaraghavan, Lyrics Inducer using Bidirectional Long Short-Term Memory Networks, Department of Computer Engineering, St. Francis Institute of Technology, Mumbai https://immohann.github.io/Portfolio/LyricsInducer.pdf

[5] Gill, Harrison; Lee, Daniel; and Marwell, Nick (2020) "Deep Learning in Musical Lyric Generation: An LSTMBased Approach," The Yale Undergraduate Research Journal: Vol. 1 : Iss. 1 , Article 1. https://elischolar.library.yale.edu/yurj/vol1/iss1/1

[6] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002) "BLEU: a Method for Automatic Evaluation of Machine Translation,"Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 311-318. https://aclanthology.org/P02-1040.pdf

```
Epoch 6000, Loss: 1.3111
=> generated Lyrics : new it s be the light on the star only  good me her up wake th
Cumulative 1-gram: 0.944444
Cumulative 2-gram: 0.577350
Cumulative 3-gram: 0.278734
Cumulative 4-gram: 0.108559

Epoch 10000, Loss: 0.7766
=> generated Lyrics : new the meder we come alive  there s still see your heart bejen
Cumulative 1-gram: 0.857143
Cumulative 2-gram: 0.574169
Cumulative 3-gram: 0.305377
Cumulative 4-gram: 0.125712
```

Figure 3: Instances of Loss and lyrics generated per 500 epochs by LSTM

```
output #Logistic Regression

'its there summer obvious should don't but a night guitar goes me star in in crazy in crazy in this in this
world'
```

Figure 4: Instances of Loss and lyrics generated per 50 epochs by Logistic Regression