Music and the Generated Word

Alex Shaw, Grant White, Walker Hughes, Johnson Merrell

April 2021

Abstract

Natural Language Processing and Music Generation are fields of machine learning that go hand-in-hand. In this project, we implement Markov models and recurrent neural networks in python and train them on song lyric and music data to create entirely new music and lyrics. Markov models with 2- to 3-dimensional state spaces outperform our RNN in text generation, largely due to overfitting leading to plagiarism. On the other hand, the RNN generates more coherent lyrics than a 1-gram Markov chain. Further, the RNN applied to song music obtained fairly high accuracy scores when predicting subsequent notes. Though our models are trained on religious hymns, our results do not have specific religious implications.

1 Problem Statement and Motivation

In 1950, Alan Turing famously developed the imitation game to challenge machines' abilities to appear human. In this test, an interviewer asks the machine questions through a text-only interface and receives answers in the same way. If the interviewer cannot discern whether or not he or she is interacting with a machine or with a human, then the machine passes the test[5].

Recent breakthroughs in programming languages and hardware acceleration have made it more feasible for a machine to pass the test. With the advent of deep learning, certain models have demonstrated human-like creative abilities. At the time of this paper, OpenAI's GPT-3 model and DeepMind's Wavenet model are the most state of the art programs for generating human-like text and music, respectively[2][4]. These models generate creative content in diverse applications.

In this paper, we likewise seek to compare different models' abilities to produce text and music. We specifically seek to design a number of models to produce hymns similar to those used by The Church of Jesus Christ of Latter-day Saints. We hope to answer the following questions: Is it possible for a machine to produce art like a human? Specifically, can a computer write understandable lyrics? Can a computer compose the music for hymns?

2 Data

For our data, we use the lyrics and instrumental music from the Church of Jesus Christ of Latter-day Saints' Hymn Book.

2.1 Data Collection

We download the instrumental soundtracks (.mp3 format) from the Church's official website. We download MIDI files for the hymns from one of Brigham Young University Idaho's websites. Finally, we scape the Church's official website for song lyrics. There are 341 songs in the English hymn book, though we do not process (scrape or download media) for 10 of the songs to avoid copyright infringement. This leaves us with a dataset of 331 songs. It is worth noting that thousands of data points come from each song (sound samples, notes, and lyrics), providing us with large training datasets (between 100,000 and 1,000,000 observations).

2.2 Data Processing

In total, we work with three datasets: lyrics, notes, and sound samples. Each dataset requires different processing.

Five steps go into processing the lyrics. First, we split each word into it's own line. Next, we remove verse numbers and replace sentence-ending punctuation and end of songs with stop words. Finally, we remove all remaining punctuation and split each word into its own line.

No data processing is necessary for the MP3 files, other than making all songs the same length through truncation.

Very little data processing is necessary for the MIDI files. We extract the chords/notes and the offsets (or space) between pitches from the MIDI files for each song in our dataset.

2.3 Data Analysis

To better understand our future models' performances, we provide a baseline against which we later compare models' outputs. Specifically, we look into the most common types of punctuation and lyrics, after removing stop words.

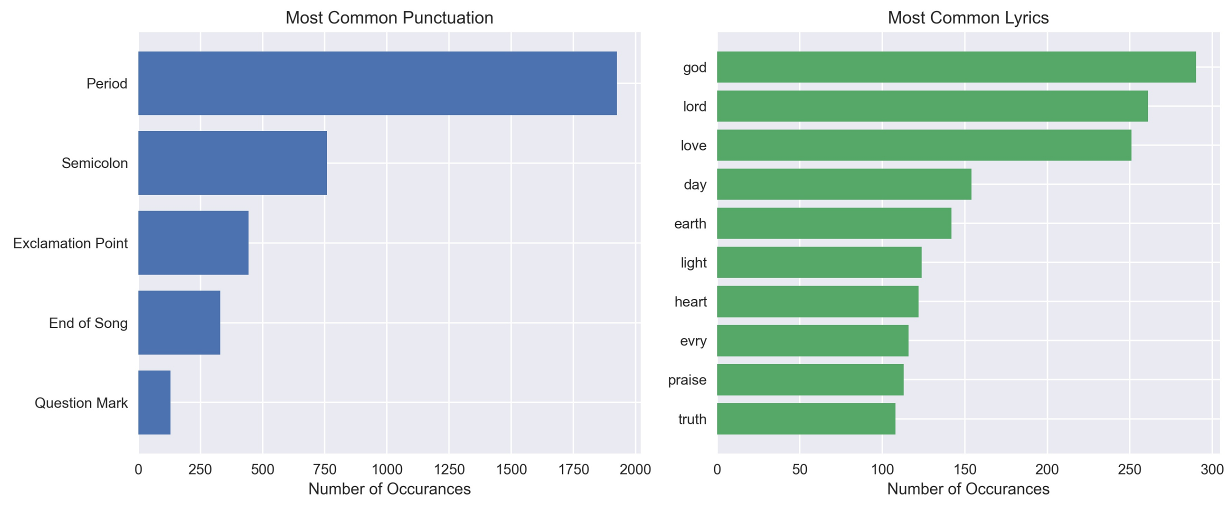

Figure 1: Most common lyrics and punctuation in LDS hymns.

As expected, the period is the most common punctuation mark, with the question mark being the least common. The three most common words used in hymns are “God”, “Lord”, and “love”.

Visualizations of the audio dataset provide little to no insight and are not included.

3 Methods & Results

3.1 Lyric Generation

We considered two methods for lyric generation: Markov chains and recurrent neural networks.

3.1.1 Markov Chain

We start by implementing a Markov chain. Our state space is the set of words and punctuation from LDS hymns. Our transition probabilities are the weighted probabilities of word x following word y in a hymn.

The benefits of using a Markov chain is the extremely quick training time and the straight forward implementation. The main drawback is the lack of long-term memory. To mildly mitigate this drawback we try using 2-grams or 3-grams as the state space. Anything larger results in text memorization. Below are excerpts of Markov chain generated hymn sentences using a 1-gram and 2-gram model, respectively.

> This through all lands the sun of our view, The foe's haughty may rise triumphant from God of Jesus died on Jesus we'll hear!
>
> In our hearts in unity, And help to you this journey may appear, Grace shall be thy supply.

3.1.2 Recurrent Neural Network

For our Recurrent Neural Network (RNN) we use a variant of the char-RNN originally developed by Andrej Karpathy[3]. This RNN creates sequences of words after being trained probabilistically on sequences of *characters*. The RNN can then be sampled to generate new sequences of text that are similar to the text it was trained on.

We trained for 500 epochs with a cross entropy loss function. Though this RNN is able to produce lyrics that are similar in style to the Hymns it is trained on, some words are unintelligible or misspelled, and grammar is often incorrect (see sample output below).

> When the sage sweet, sing, that shall love the sweet Then sinless prophets lead on earth and faith, With power, their favors peace.
>
> Let lend the day and chosen at home Thy name our prayer, your love and these We brings us to greet

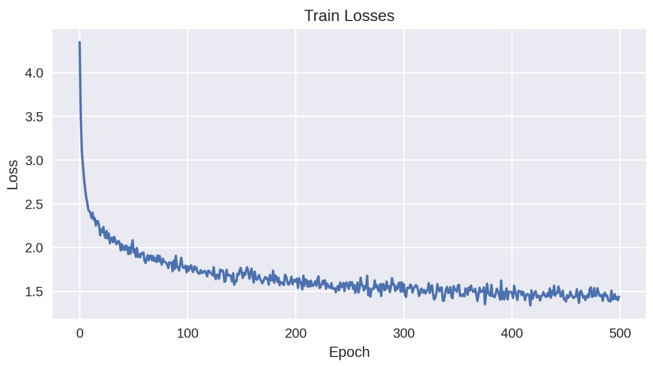

Figure 2: RNN Training Losses over 500 Epochs.

3.2 Audio Generation

We likewise consider three similar methods for *audio* generation: Markov chains, recurrent neural networks, and 1-dimensional convolutional deep generative neural networks.

3.2.1 Markov Chain

Using a similar approach to that of the Markov chain for lyric generation, we generate audio using a single-state Markov chain, a 2-gram, and a 3-gram. We use the note/chord data from the MIDI files in our Markov chain to predict the next note from previous ones. The offset between notes is all set to the same value: 0.5.

Markov chains have a quick training time and straight forward implementation. The main drawback is the lack of long-term memory. Even the use of 3-grams only uses the previous three notes to predict the next note.

3.2.2 Recurrent Neural Network

Using MIDI files, we train an RNN on audio generation. Our dataset is the sequence of notes in LDS hymns. In our RNN, we use three LSTM's and two linear layers, along with dropout and batch normalization to prevent overfitting. Our objective function is cross-entropy loss. We train with sequences of 30 notes to take advantage of long-term memory and avoid sequence lengths similar to that of a hymn. The offset between notes was all set to the same value: 0.5. The training loss and accuracy curves are shown below.

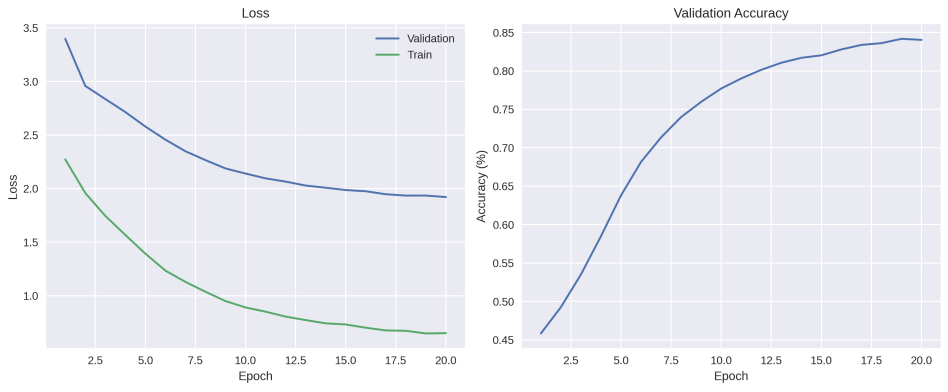

Figure 3: RNN train losses and accuracy.

The benefit of using an RNN (with LSTM's) is the long-term memory. This makes it a natural choice to model music, which relies greatly on the previous chords and notes. In many scenarios, RNN's are considered state-of-the-art for sequential modeling. The downside of using an RNN is the long training time. Sequential models can not train effectively in parallel.

3.2.3 1-D Generative CNN (DeepMind's Wavenet)

Aäron van den Oord describes Wavenet as a "deep generative model of raw audio waveforms." It is extremely difficult for programs to track and reproduce local and global patterns found in music, but Wavenet's unique use of 1-dimensional convolutional layers performs this task exceptionally. Wavenet's architecture enables it to receive raw audio (in the form of .wav files) as input, rather than first having to perform expensive feature engineering. Wavenet predicts a single sample at a time, with 16,000 samples amounting to one second of audio[4]. This makes audio generation prohibitively slow.

We use Igor Babuschkin's TensorFlow implementation of Wavenet[1] and train it with .wav-format audio files of each hymn. Iterating through all 331 .wav files defines a single epoch. After training the model for the recommended number of epochs, we find training loss to decrease by 73%. We then generate 10 seconds of audio and find the results to be extremely disappointing. The audio was mostly white noise, with some minor changes in pitch. Because of how poor it performed, we exclude it from further analysis.

4 Analysis

4.1 Lyrics

For our Markov chain model, we compare results from n-gram models, for $n \in \{1, 2, 3\}$. We check for percent of output plagiarized from LDS hymns (using desklib.com), as shown in the bar chart below. Clearly using 2- or 3-gram Markov chains results in over-fitting. Clearly, the 1-gram Markov chain produces the most unique, albeit meaningless, lyrics. On the other hand, the 2-gram Markov chain produces coherent, meaningful lyrics at the expense of large amounts of plagiarism.

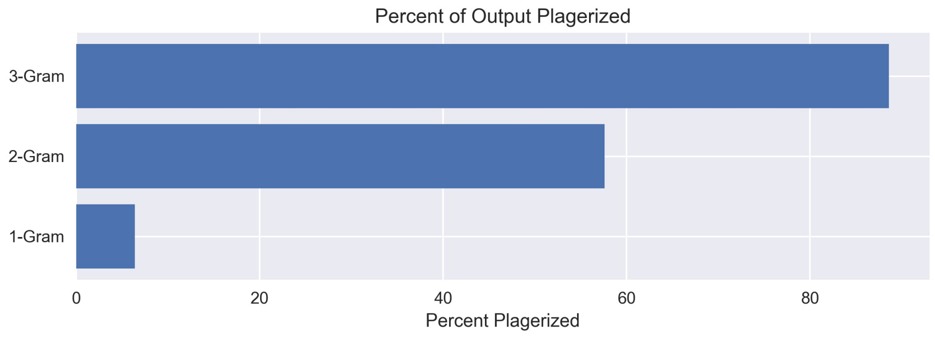

Figure 4: Average percentage of output plagiarized from LDS Hymns.

As seen in §3.1.2, our RNN does a fairly good job of producing understandable text. Its main difficulty is with spelling rather than coherency. In addition to our qualitative assessment, we examine the word and punctuation distributions of the RNN output.

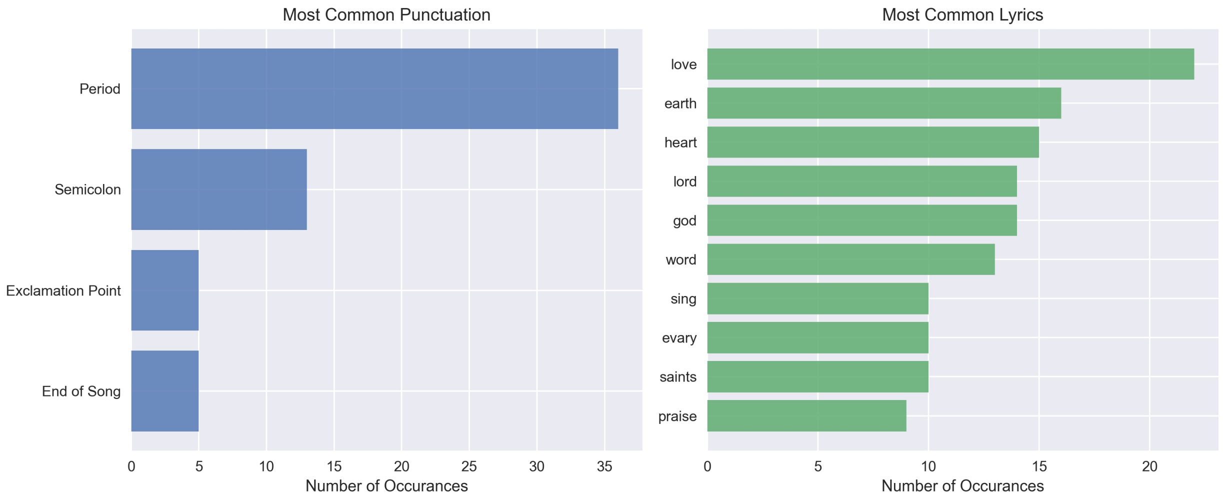

Figure 5: Most common lyrics and punctuation in RNN hymns.

Interestingly, six of the RNN's ten most commonly used words also appear in the top ten words of LDS hymns. Punctuation-wise, we see similar results, with periods and semi-colons being used at similar rates to that of LDS hymns. This demonstrates similar themes and flow between our generated lyrics and those of real hymns.

4.2 Music

Out of all the Markov chain models, the 3-gram does the best at producing music. The music produced by the 1-gram is basically random notes. The 2-gram produces short intervals that sound good, but the overall hymns do not sound cohesive. Though not passable for an actual hymn, the 3-gram does a good job producing music. The drawback is that there isn't a consistent global pattern for the hymns. It is also interesting to note that the Markov model chooses mostly notes instead of chords.

Though the RNN is a little repetitive in the chords that it chooses, it performs adequately at producing hymns. Unlike the Markov models, it chooses a lot of chords as well as some notes.

For both the RNN and Markov chains, we choose offset to be constant for MC and RNN. Drawing randomly from the distribution of offsets does not produce good results because the space between notes shouldn't be random. Offsets could instead be produced by their own Markov chain or RNN.

5 Ethical Considerations

The most prominent ethical consideration is the legal use of lyrics and music from the Church of Jesus Christ of Latter-day Saint's hymnbook. After careful research, we confirm that 331 of the 341 songs are permitted to be downloaded. To comply with these restrictions on the 10 songs, we also do not scrape their lyrics. The MIDI files are all publicly available on a website from Brigham Young University Idaho.

Further, when implementing bi-gram and tri-gram Markov chains on hymn lyrics, we found that the models overfit the lyrics they were trained on, leading to plagiarism risk. Many of these lyrics were essential copies of true LDS Hymn lyrics. With the lyrics generated from the tri-gram model, we obtained on average an 88% plagiarism score from desklib.com, an online plagiarism checker. In this sense, overfitting may lead to plagiarism, which is of course unethical. When implementing Markov models for text generation, it is advisable to not use a tri-gram in order to avoid implicit plagiarism.

The other, less well-defined ethical consideration is the use of religious text for machine learning. We do not intend to compromise the religious integrity of the hymns through their use in our project. As a safety precaution, we do not make any generated text or music publicly available, and specifically state that our generated text does not have religious implications.

6 Conclusion

While the bi-gram language model produces the most coherent lyrics, we cannot accept it to be the best model because of its tendency to plagiarize lyrics. Consequently, we find that the RNN produces the best unique lyrics. Unfortunately, none of the language models produce any lyrics that are both unique and satisfactory.

The RNN and the 3-gram Markov chain perform the best at music generation. The RNN is more likely to include chords while the 3-gram Markov chain has more interesting note patterns. The more complicated Wavenet produces white noise almost exclusively. Overall, the music produced by the RNN is the closest to resembling a traditional LDS Hymn.

Future research should explore transformers for lyric generation. Efforts hereafter should also modify the state-spaces for RNN's and Markov chains to include both pitches and their offsets.

References

[1] Igor Babuschkin. *tensorflow-wavenet*. https://github.com/ibab/tensorflow-wavenet. Accessed: 2021-04-09.

[2] Tom B Brown et al. "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165* (2020).

[3] Andrej Karpathy. *char-rnn*. https://github.com/karpathy/char-rnn. Accessed: 2021-04-08.

[4] Aäron van den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[5] Alan M Turing. "Computing machinery and intelligence". In: *Parsing the turing test*. Springer, 2009, pp. 23–65.