Generating Melody for lyrics

Damanpreet Kaur
College of EECS, Oregon State University
1148 Kelley Engineering Center
Corvallis, OR 97330
kaurd@oregonstate.edu

Prachi Rahurkar
College of EECS, Oregon State University
1148 Kelley Engineering Center
Corvallis, OR 97330
rahurkap@oregonstate.edu

Abstract

In this project, we aim to develop a model that learns to compose music for natural language. The model is provided with poetry lyrics as input and generates a suitable melody for these lyrics as output. The model consists of two LSTM layers followed by a dense layer with a SoftMax activation function. The model was trained on a pair of lyrics-melody of 532 songs which depicts mixed emotions. The dataset comprises of sad and happy songs. The results of the experiment show that the LSTM model was successfully able to generate a melody which captures the sentiment of the lyrics.

1. Introduction

For the experiments in this study, we chose a dataset which comprises of multiple sentiments. The model takes song lyrics as input and generates a melody for it as a result. The main aim is that the model should learn to capture the sentiments of the text and generate a melody for it. We chose ABC music notation for music as it is easy to understand compared to the other music notations. In ABC notation, the music is just a sequence of characters. We feed these characters in sequence to our model at each time step along with the words from the lyrics. Since our output is a multiclass classification problem, so we use cross entropy as our loss function. The model not only learns the patterns of the musical notes from our training data but also learns the emotion in the lyrics.

We collected 532 lyrics-melody pairs to carry out our experiment. Though these songs capture multiple sentiments, due to the lack of knowledge of music we broadly classify the testing lyrics into sad and happy moods and carried out human evaluation based on it. The human evaluation results further show that the generated tones were able to convey the sentiments of the song. Though the generated melody was not pleasing, it was still able to capture the emotions which was the aim of our experiment.

Novelty: The factor of novelty in this project is the emotion analyzing part where the model learns to generate music according to the sentiment of the input lyrics. Much research has been carried out to generate novel music. But, to the best of our knowledge, generating music for the input lyrics is an ongoing research area and no solution has yet been established to solve this problem.

1.1. Concepts from music theory

Melody can be regarded as an ordered sequence of many musical notes. The basic unit of melody is the musical note which mainly consists of two attributes: pitch and duration. The pitch is a perceptual property of sounds that allows their ordering on a frequency-related scale; or in simple words, the pitch is the quality that makes it possible to judge sounds as “higher” and “lower” in the sense associated with musical melodies. Duration is a time interval to describe the length of time that the pitch or tone sounds, which is to judge how long or short a musical note lasts.

An ABC file[1] (shown in figure 1) in music consists of a tune header and tune body, terminated by the end of file tag. The tune header consists of metadata. The tune header should start with a X: and end with a K: field. The tune body follows the header and contains the music code. It consists of notes, bar lines, and other musical symbols. Chords are coded with [ ] symbols around the notes. All the notes within a chord mostly have same length, but if that’s not the case, the

[1] ABC File format http://abcnotation.com/wiki/abc:standard:v2.1

chord duration should be equal to that of the first note. Example – [CEGc] indicates the chord of C major.

```
<score lang="ABC">
X:1
T:The Legacy Jig
M:6/8
L:1/8
R:jig
K:G
GFG BAB | gfg gab | GFG BAB | d2A AFD |
GFG BAB | gfg gab | age edB |1 dBA AFD :|2 dBA ABd |:
efe edB | dBA ABd | efe edB | gdB ABd |
efe edB | d2d def | gfe edB |1 dBA ABd :|2 dBA AFD |]
</score>
```

Figure 1: ABC notation[2]

1.2. Task Definition

Given lyrics as the input, our task is to generate a suitable melody for it. We can formally define this task as below:

The input is a sequence of words $X = (x^1, x^2, \ldots, x^{|X|})$ in the lyrics. The output is a sequence of musical notes $Y = (y^1, y^2, \ldots, y^{|Y|})$ for the lyrics.

2. Related Work

A variety of music composition works have been done over the last decades. Most of the traditional methods compose music based on music theory and expert domain knowledge. Chan, Potter, and Schubert (2006) design rules from music theory to use music clips to stitch them together in a reasonable way. With the development of machine learning and the increase of public music data, data-driven methods such as Markov chains model (Pachet and Roy 2011) and graphic model (Pachet, Papadopoulos, and Roy 2017) have been introduced to compose music.

Recently, deep learning has revealed the potentials for musical creation. Most of the deep learning approaches use the recurrent neural network (RNN) to compose music by regarding it as a sequence. The MelodyRNN (Waite 2016) model, proposed by Google Brain Team, uses Look-back RNN and Attention RNN to capture the long-term dependency of a melody. Chu, Urtasun, and Fidler (2016) propose a hierarchical RNN based model which additionally incorporates knowledge from music theory into the representation, to compose not only the melody but also the drums and chords. Some recent works have also started exploring various generative adversarial networks (GAN) models to compose music (Mogren 2016; Yang, Chou, and Yang 2017; Dong et al. 2017).

3. Approach

We conducted our experiments on two sequential models, which use LSTM and GRU. We found that LSTM is a better fit for our task.

3.1. Overview of LSTMs

LSTM network, which stands for Long Short-Term Memory, is a type of recurrent neural network. LSTMs help to preserve the error for a long time which can be backpropagated through time. They are capable of learning order dependence in sequence prediction problems and address the problem of vanishing and exploding gradients in RNN. In simple words, LSTM networks have some internal contextual state cells that act as long-term or short-term memory cells.

The output of the LSTM network is modulated by the state of these cells. This is a very important property when we need the prediction of the neural network to depend on the historical context of inputs, rather than only on the very last input. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at any cell state has three different dependencies.

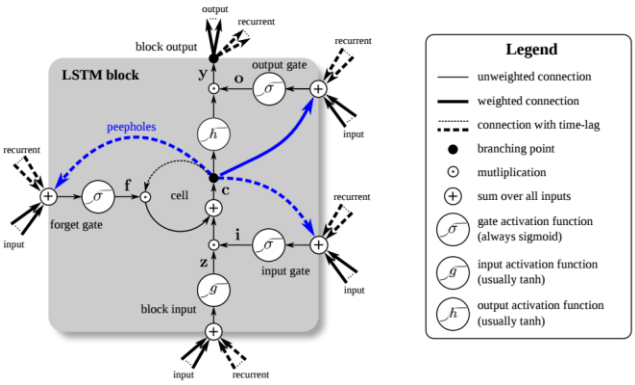


Figure 2: A Long Short-Term Memory (LSTM) unit. The LSTM unit has four input weights (from the data to the input and three gates) and four recurrent weights (from the output to the input and the three gates).

These dependencies can be generalized to any problem as:

[2] https://en.wikipedia.org/wiki/ABC_notation

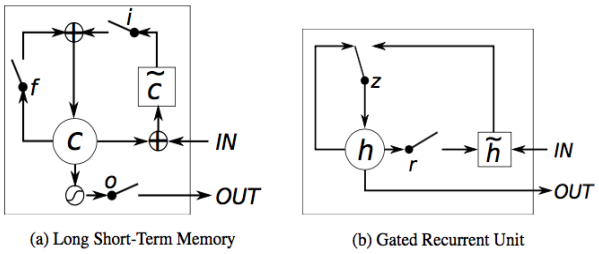
1. The previous cell state (i.e. the information that was present in the memory after the previous time step).
2. The previous hidden state (i.e. this is the same as the output of the previous cell).
3. The input at the current time step (i.e. the new information that is being fed in at that moment).

A forget gate (as shown in Figure 2) is responsible for removing information from the cell state. The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter. This is required for optimizing the performance of the LSTM network. This gate takes in two inputs; h_{t-1} and x_t: h_{t-1} is the hidden state from the previous cell or the output of the previous cell and x_t is the input at that time step. The input gate is responsible for the addition of information to the cell state. The job of selecting useful information from the current cell state and showing it out as an output is done via the output gate.

3.2. Overview of GRU

Gated recurrent units are a gating mechanism in recurrent neural networks. GRUs are similar to LSTMs. But there are some differences between LSTM and GRU which are worth noting -

1. GRU is a long short-term memory with forget gate but no output gate (as shown in figure 3). GRU has an update and reset gate. Therefore, it has less parameters as compared to LSTM.
2. GRUs don't possess any internal memory that is different from the exposed hidden state.
3. There is no second non-linearity applied when computing the output in GRU. LSTM, unlike GRU, has a sigmoid before the output gate.



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 3: Illustration of (a) LSTM and (b) Gated recurrent units. (a) i, f and o are the input, forget and output gates, respectively. c and c˜ denote the memory cell and the new memory cell content. (b) r and z are the reset and update gates, and h and h˜ are the activation and the candidate activation.

GRU is more efficient than LSTM as it trains the model faster than LSTM and may need less data to generalize. Though the performance of GRU is comparable to LSTM, LSTM outperforms GRU in many situations. When long term memory is required, and enough data is available, LSTM can lead to better results at the expense of more computation cost.

4. Experiments Tried/ Phases

4.1. Dataset Collection

Selecting the right music format was a challenge. Most of the early research in generating music use MIDI files for their experiments. We obtained a dataset for midi-lyrics pair as we initially started working on midi files. This dataset consists of 50 songs.

We eventually chose to work on the ABC file format since this format of music was easy for us to understand as it is a sequence of characters. We crawled 532 songs from the website www.lotro-abc.com. We then pre-processed the data to exclude the missing and invalid entries. Removing invalid entries from the dataset needed manual intervention. We then pre-processed the dataset more to remove the meta- information from the files like title (T:), when there is more than one tune in a file (X:). We use a validation split of 0.20 and randomly shuffle the songs to obtain training and validation datasets. We work on 425 songs for training and 107 songs for validation. We chose test lyrics of 10 songs to generate melodies for carrying out human evaluation.

4.2. Implementation (Model Architecture)

Phase 1: Using LSTM

The implementation of this experiment was carried out using the model shown in Figure 4. It consists of two LSTM layers stacked one on top of the other. We added a layer of dropout (dropout = 0.3) after every LSTM layer. Because of multiple LSTM layers, more complex input patterns can be described at every layer allowing greater model complexity. Adding dropout

enables the network to learn multiple independent internal representations. The effect is that the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data.

A dense (linear) layer is added on top of these layers, with a SoftMax activation function. The output prediction at timestamp (t-1) is fed to the next cell of timestamp (t).

The input to our model is the concatenated vector consisting of the text embedding (of the lyrics) and the melody embedding (of the musical notes). The output generated by the model is a continuous sequence of notes (melody) for the corresponding input lyrics. This output is in the format of an ABC notation which can be played using any online ABC-to-MP3 or ABC-to-MIDI converter.

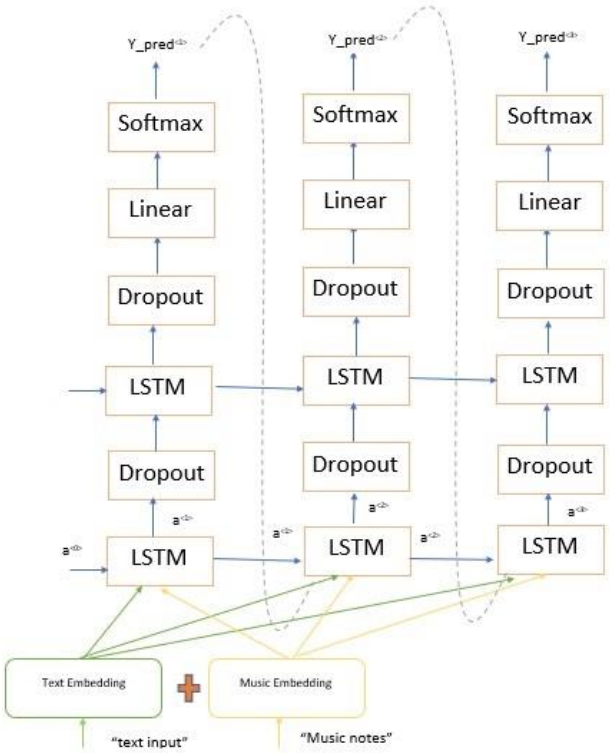


Figure 4: Model architecture used in the prediction of musical notes

Training: We use Adam as the optimizer with an initial learning rate of 0.001 and batch size of 1 to train our model, and the cross entropy as the loss function. The network was able to learn from the text and music sequence fed to it as input during training.

In figure 5, we show that the loss curve of the best validation run using LSTM. We see that the loss drops quickly in the first few epochs and drops slowly in the later epochs.

Figure 5: Loss curve (Loss per epoch) for training and validation dataset using LSTM.

Phase 2: Using GRU

We tried using GRU instead of LSTM. Given the lyrics and its corresponding melody as input, we find the text embedding of the lyrics and embedding of the melody. It encodes the lyrics and melody into two vectors which are fed as an input to the GRU. The text embedding of one word and music embedding of one note is passed as an input to the GRU at each time step. A dropout of 0.3 was added to this layer. This output was then fed into another GRU layer with a dropout of 0.2. A dense layer with Softmax activation was stacked on the top of these layers. The output of each time step was fed as an input to the next timestep. We observed that GRU degraded the performance of the network and none of the generated melodies sounded good to humans.

4.3. Results

Since we did not have any baseline model to compare our results with, we chose to get our results evaluated by humans. We took 10 song lyrics and generated melodies for these lyrics using our model. We then asked the humans to listen to the obtained melodies and categorize them into sad and happy tones. We made sure that the evaluators did not look at the corresponding lyrics or lyrics label during the study. Below is the summary of the obtained results:

For 8 songs the label category of the generated melodies matched the actual lyrics category i.e. if the tone of the lyrics was sad, the generated melody was also categorized as sad by the human evaluators.

Here is the track we obtained for the lyrics of a song which has a happy emotion: https://soundcloud.com/damanpreet-kaur-9/coldplay-scientist. Here is the track we obtained for the lyrics of a song which has a sad emotion: https://soundcloud.com/damanpreet-kaur-9/we-the-king.

The generated melodies do not sound like an original piece of music, but they were still able to capture the sentiment of the lyrics.

5. Conclusion

The contributions of our work in this report are summarized as follows:

- To the best of our knowledge, our experiment is the first work to compose melody corresponding to input lyrics (considering the context of the input text).
- We construct a lyrics-melody dataset consisting of 532 songs.
- The human evaluation verifies that the synthesized musical notes of the generated melody are meaningful.

6. Future work

We are interested in trying out the encoder-decoder architecture. We believe that by using this architecture, our model will be more robust and will generate melodious tones. The sequence to sequence architectures have shown success in music generation. We would also like to incorporate attention mechanism where the model will use attention to look at all the encoder outputs at each step in the decoder. Using this, our model will be able to learn to attend to different parts at each step of the decoder. We also wish to work with a dataset which consists of melodies played on multiple instruments and produce more varied tunes.

7. References

[1] Yang, L.-C.; Chou, S.-Y.; and Yang, Y.-H. 2017. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. arXiv preprint arXiv:1703.10847.

[2] Roberts, A.; Engel, J.; Raffel, C.; Hawthorne, C.; and Eck, D. 2018. A hierarchical latent vector model for learning long-term structure in music. arXiv preprint arXiv:1803.05428.

[3] Mogren, O. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. arXiv preprint arXiv:1611.09904.

[4] Hasim Sak, Andrew Senior, Francoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. arXiv preprint arXiv: 1402.1128.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv preprint arXiv: 1412.3555.

[6] Chan, M.; Potter, J.; and Schubert, E. 2006. Improving algorithmic music composition with machine learning. In Proceedings of the 9th International Conference on Music Perception and Cognition, ICMPC.

[7] Pachet, F., and Roy, P. 2011. Markov constraints: steerable generation of markov sequences. Constraints 16(2):148–172.

[8] Waite, E. 2016. Generating long-term structure in songs and stories. Magenta Bolg.

[9] Pachet, F.; Papadopoulos, A.; and Roy, P. 2017. Sampling variations of sequences for structured music generation. In Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR'2017), Suzhou, China, 167–173.

[10] Dong, H.-W.; Hsiao, W.-Y.; Yang, L.-C.; and Yang, Y.-H. 2017. Musegan: Symbolic-domain music generation and accompaniment with multi-track sequential generative adversarial networks. arXiv preprint arXiv:1709.06298.

[11] Chu, H.; Urtasun, R.; and Fidler, S. 2016. Song from pi: A musically plausible network for pop music generation. arXiv preprint arXiv:1611.03477.