Finding an Embedding for Music Auto-Complete: An LSTM Approach

J.N. Andrews Honors Program
Andrews University

Nathaniel Patterson
Rodney Summerscales, PhD
Department of Computing
School of Business Administration, Andrews University

Abstract:

Recent success in sequence generation has been seen in the problem of Polyphonic Music Generation. This study analyzes the effectiveness of two embedding strategies: notes as a string and notes as objects using a Long-Short Term Memory Recurrent Neural Network (Hochreiter 1997) for music auto-completion when trained on the corpus of Erik Satie. This project seeks to introduce Music Autocomplete as a new problem, while adding to the body of knowledge on how Neural Networks process sequential data and how different data embeddings improve performance. This project also adds to the subfield of the intersection of artistry and Artificial Intelligence.

Figure 1: LSTM Model Summary

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 100, 256)          264192
_________________________________________________________________
dropout_3 (Dropout)          (None, 100, 256)          0
_________________________________________________________________
lstm_4 (LSTM)                (None, 100, 512)          1574912
_________________________________________________________________
dropout_4 (Dropout)          (None, 100, 512)          0
_________________________________________________________________
lstm_5 (LSTM)                (None, 256)               787456
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 19)                4883
_________________________________________________________________
activation_1 (Activation)    (None, 19)                0
=================================================================
Total params: 2,697,235
Trainable params: 2,697,235
Non-trainable params: 0
```

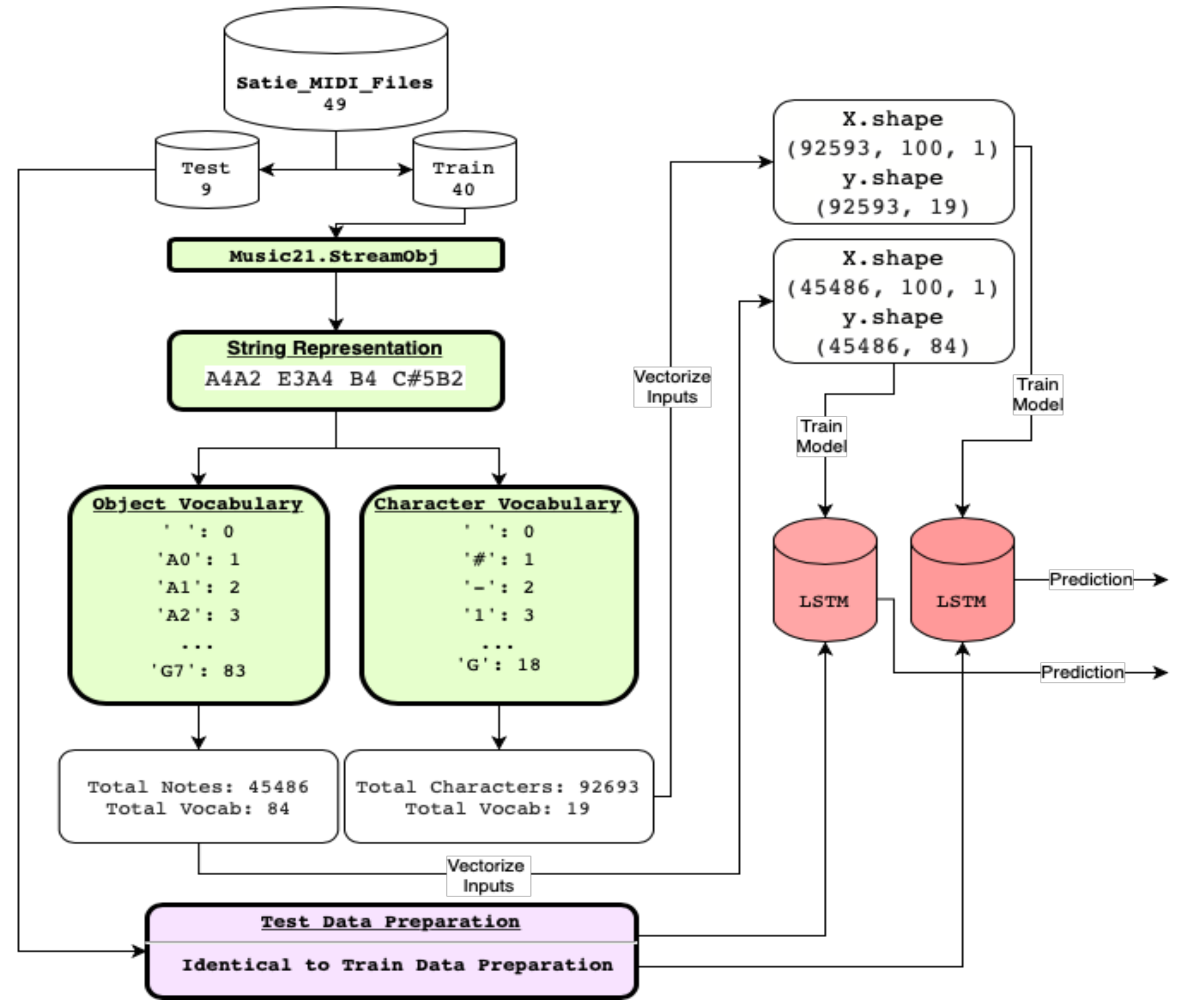
Methodology:

1. **Select Artist** Erik Satie was the artist of choice for this project for his well known piano works and recognizable style.
2. **Gather Data:** We identified 157 piano compositions and were able to locate the MIDI files for 76 of them (roughly 48%). This includes 9 "major works."
3. **Prepare Data:** We used the Music21 python library for MIDI file data mining. Data were split into training and testing sets, and then using the Music21 modules, the songs were translated into both data representations. Once in the respective representation, we used a sequence of 100 characters for the string model and 70 notes for the object approach. Then the data were prepared for the model.
4. **Build Model:** The simple LSTM-RNN model used consists of three LSTM layers with dropout 0.3 and 512, 256, and 256 nodes per layer respectively and two dense layers one with 256 nodes and the other with node count equal to the length of the output vector.
5. **Train the Model:** Models were tracked based on their accuracy during training on the validation split set. We used a timeseries split with n=5 to avoid overfitting. Best weights based on validation loss were saved using early stopping and checkpoints.
6. **Evaluate the Model:** Using the test data split in step 3, we predict the top 5 suggested next notes in the sequences and analyze the results qualitatively.

Data:

We selected Erik Satie and the MIDI file format for this research due to the extensive libraries of classical MIDI files. We identified 157 piano compositions and were able to locate the MIDI files for 76 of them (roughly 48% of total works). This includes 9 of 10 "major works." Many works have multiple movements, which were combined into the same file in cases of movements with short total note sequences. Due to this, our file count is 49. We then randomly split the dataset into 40 file training and 9 file testing set. We decided that given the nature of this problem and the available data, that restricting music from particular time periods would not be necessary. Further, we only wish to predict accurately what note(s) were played at the next timestep, so no attention was given towards melodies and dynamics.

Figure 1: Data Flow for Respective Embeddings



Results and Analysis:

During the training process for the character-based LSTM we saw overfitting on the validation data at around 8 epochs during each fold in the time series split. So the model was trained for 8 iterations on the entire training dataset. The character LSTM model had the most success in autocompletion.

Table 1: Average Loss and Accuracy for Character Model

Character Based LSTM Average Accuracy			
	Trial A	Trial B	Trial C
train_acc	47.88%	51.26%	44.61%
val_acc	36.94%	37.37%	37.00%

The LSTM framework is much better suited to the character approach than to the object approach. Over three independent trials on the nine test files, we saw a 16.7% success rate. Success rate was determined by whether or not the true note occurred in the top five suggestions list from the data. Table 1 shows a sample output. Further, while if let train for a while, the training set could reach 99% accuracy with <5% loss, however, any validation accuracy could not rise above around 45% before overfitting.

The results from the object-based LSTM were not reportable, as the model was unable to train on the data.

Table 2: Model Output Example

```
satgno01.mid
F2 F4C4C5G#3 E-5 D5 F4C4C5G#3 B4F2 F4C4G#3 C5 B4 F4C4G#3 F2 F4C4C5G#3 E-5 D5
F4C4C5G#3 E5 F5F2 F4C4G

['#3 ', '3 ', '4B3 ', '5A3 ', '6A3 ‘]

success: ‘#3’
```

```
Enfantillages_Pittoresques2.mid
D5F5A4 C5 F5A5B4 C5 E5G5A4 C5 F5A5B4 C5 D5F5A4 C5 F5A5B4 C5 E5G5A4 C5 F5A5B4 C5
E5G4B4 D5 E5F4A4 D5
['A4E4 ', 'B4E4 ', 'D5G4A4 ', 'C4E4 ', 'F5G4B4 ‘]

correct note/chord is: ‘E5G4B4’
```

Discussion & Conclusion:

The original intent of this project was to implement two embedding strategies on one network so that performance could be compared. However, in the end this reinforced the truth that there are models better suited to certain embedding strategies, and while the best embedding from a performance standpoint for a simple LSTM model is the character-based model, there is much to discover in lieu of architecture for the object approach.

It is worth noting that the large vocabulary of the object based model became a huge detriment due to the imbalanced appearance of the space character when dividing timesteps. This made it very hard for the model to train even when class weights were balanced. Thus a true comparison between embeddings was not fair, as a different model architecture would likely improve performance. Better yet, a data representation that would take a vector at each timestep and then have an output vector of many 1s would show the relationship between notes more accurately. This however fits into the category of multiple output classes which has seen better success with Restricted Boltzmann Machines (RBMs) rather than LSTMs. While the results for the object approach were insignificant, further research is warranted into the application of RBMs and RBM-RNNs to this problem. There have also been successful models in the polyphonic music generation problem such as the Bi-Axial LSTM (Johnson 2017), so an evaluation of how well the Bi-Axial LSTM can autocomplete an artist's work would also be interesting.

While we saw limited success in our character representation, there is much more to be gained by experiments with cutting edge architecture. Other key limitations of this study are that the method for determining success is tough to quantify. Further, music was selected as it was available. More research could be done with better access to MIDI files of Erik Satie, or a comprehensive list of some other artist. While this study makes an effort to express how well the model is doing at autocompleting, it is somewhat unfair to say whether or not it has done well. One of the takeaways from this research is that further interdisciplinary study is warranted to develop a good metric for music autocomplete analysis, given that the model could suggest a note or a chord that is technically similar to the true chord, though it is not the exact notes played at that time. Once a good evaluation system is developed, then more results can be compared across architectures and embeddings.

Works Cited:

Johnson, Daniel D. "Generating Polyphonic Music Using Tied Parallel Networks." Computational Intelligence in Music, Sound, Art and Design Lecture Notes in Computer Science, 2017, pp. 128–143., doi:10.1007/978-3-319-55750-2_9.

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long Short-Term Memory." *Neural Computation*, vol. 9, no. 8, 1997, pp. 1735–1780., doi:10.1162/neco.1997.9.8.1735.

Acknowledgements: Special thanks to Musicalion Midi Library for allowing free use of their midi files for this project