Text-based LSTM networks for Automatic Music Composition

CHOI, K; sandler, M; fazekas, G; Conference on Computer Simulation of Musical Creativity

arXiv record http://arxiv.org/abs/1604.05358. Presented at Conference on Computer Simulation of Musical Creativity

Text-based LSTM networks for Automatic Music Composition

Keunwoo Choi, George Fazekas, and Mark Sandler *

The Centre for Digital Music, Queen Mary University of London
{keunwoo.choi, g.fazekas, mark.sandler}@qmul.ac.uk

Abstract. In this paper, we introduce new methods and discuss results of text-based LSTM (Long Short-Term Memory) networks for automatic music composition. The proposed network is designed to learn relationships within text documents that represent chord progressions and drum tracks in two case studies. In the experiments, word-RNNs (Recurrent Neural Networks) show good results for both cases, while character-based RNNs (char-RNNs) only succeed to learn chord progressions. The proposed system can be used for fully automatic composition or as semi-automatic systems that help humans to compose music by controlling a diversity parameter of the model.

Keywords: LSTM, RNN, automatic composition, chord progressions

1 Introduction

Music composition is considered creative, intuitive and therefore inherently human. Nevertheless, it has a long history of mathematical approaches since Hiller and Isaacson proposed to use *Markov* chains for automatic composition [6]. The field of automatic composition includes a wide range of tasks such as the composition of melody, chord, rhythm [10], and even lyrics [3], i.e. every typical components of music, and has been subject to numerous research studies. There are many applications for automatic composition too; automatic background music generation, AI-assisted composition systems and improviser software[1] for example.

Music can be represented as a sequence of events and thus it can be modelled as conditional probabilities between musical events. For example, in harmonic tracks, some chords are more likely to occur than others given the previous chords, while the whole chord progressions often depend on the global key of the music. In many automatic composition systems, these relationships are simplified by assuming that the probability of the current state $p(n)$ only depends on the probabilities of the states in the past $p(n-k)...p(n-1)$. A sequence of musical events - notes, chords, rhythm patterns - is generated by predicting the following event given a seed sequence.

* This paper has been supported by EPSRC Grant EP/L019981/1, Fusing Audio and Semantic Technologies for Intelligent Music Production and Consumption.

[1] `http://jukedeck.com`, `http://arpegemusic.com`, *Band-in-a-Box, PG Music Inc.*

Hidden-Markov models (HMMs) are one of the most popular methods to model and predict sequences. HMMs are based on the assumption of $k = 1$ (Markov assumption) given the sequence of the hidden states which determine the visible states. Choral harmonisation is generated after learning chorales by Bach using a HMM in [1], where 229 and 153 chorales are used for training and testing, respectively. In [14], chord progressions are generated to accompany a melody to help non-musicians to create music using a HMM. The training set of the HMM consists of 298 lead sheets including pop, rock, R&B, jazz, and country music. In the prediction, the system generates chords using a 62×62 chord transition probability matrix. In practice, HMMs had been the most suitable for time-series modelling given the data, computing power, and feasible optimisation strategies. One of the drawbacks of HMMs, however, is the inefficiency of *1-of-K* scheme of its hidden states. The memory of HMM is limited to $log_2(N)$ bits when there is N hidden states, which requires to learn N^2 parameters for the transition matrix.

Recurrent Neural Networks (*RNNs*) allow for incorporating long term dependency in the model. *Jordan net* [8], a simple version of RNNs, is used in [12] to generate chord sequences. In [13], melodies were generated by a system named *CONCERT*, which is trained on sets of 10 Bach pieces to generate melodies by note-wise prediction. One ability *CONCERT* lacks is to learn the global structure; this may be due to the difficulty of training an RNNs. Theoretically, it can remember infinitely long sequences, although in practice it is limited by the *vanishing gradient* problem [7]. During the training of back-propagation through time, the gradient is extremely diminished by multiplications of sigmoid operations.

LSTM (Long Short-Term Memory) units solved this vanishing gradient problem [7]. LSTM allows the gradient to be flowed by a separate path with not multiplication but *addition* operations. LSTM is adopted in [4] to learn 12-bar Blues chords progressions and melodies. [11] focuses on the generation of percussive tracks using LSTM network. The network in [11] directly analyses audio content of drum tracks and learns features using LSTM.

In this paper, we introduce applications of character- and word-based RNNs with LSTM units for the automatic generation of jazz chord progressions and rock music drum tracks. Our work is differentiated from previous works by two aspects. First, the LSTM networks we use are designed to learn from text data rather than representations of musical symbols or numeric values. Directly using text data minimises the overall design procedures for the encoding-decoding scheme and the network. Second, compared to the previous research [1],[4],[14], the LSTM networks is trained using a large dataset, which enables itself to learn more complex relationship between the chords in a large set.

In the Section 2, we introduce character-based RNNs and the proposed architecture. In Sections 3 and 4, two case studies on the applications of RNNs to automatic composition are explained - for jazz chord progressions and rock music drum tracks. We conclude the work in Section 5.

2 The architecture

2.1 Character-based RNNs

Char-RNNs are RNNs with character-based learning [15], which is different from the conventional approach of word-based learning. When applied to the texts of chords, a char-RNN predict a vector that corresponds to a character (e.g. predict *a* based on *C:m*, and predict *j* based on *C:ma*), while a word-RNN predicts a vector, which corresponds to a unique chord (e.g. *C:maj* based on *G:maj*). Using char-RNNs in this work has two merits.

First, it is based on the minimal assumption - there is no constraint on the form of the text representation of music. It is worth inspecting if RNNs can learn musical information with such a weak assumption.

Second, fewer number of characters means fewer number of states, which results in reducing the computational cost. From a linguistics point of view, sequence learning methods such as HMMs and RNNs used to model each *word*(e.g. chord) as a *state* as it is natural to find the relationships between words. One drawback of word-based learning is the large number of states (or the size of vocabulary); in natural language processing tasks, the vocabulary size easily exceeds few thousands to even few millions. In the proposed method the size of the chord vocabulary is 1,259. With character-based prediction, this decreases to 39.

The price of small vocabulary size is a longer sequence; as we need to learn character by character, the model should *remember* a longer sequence of states. As mentioned above, the LSTM unit helps the RNNs to learn this long-term dependency better. This trade-off does not necessarily benefit as in Section 4.

2.2 The Proposed Architecture

We use two LSTM layers, each of which consists of 512 hidden units. Dropout of 0.2 is added after every LSTM layers [16].

We use the *Keras* deep learning framework [2]. During the optimisation, categorical cross-entropy is used as a loss function and optimisation is performed by ADAM [9]. This optimiser shows an equivalent final performance to Stochastic Gradient Descent with Nestrov momentum with faster convergence.

The prediction is stochastic. In each prediction for time index n, the network outputs the probabilities of every states. To make the system *tunable*, We employ a diversity parameter α in the prediction stage (see Eqn. **??**), which suppresses ($\alpha < 1$) or encourages ($\alpha > 1$) the diversity of prediction by re-weighting the probabilities. In detail, the probabilities of i-th state, p_i, are re-weighted as $\hat{p_i} = \exp(\log(p_i)/\alpha)$. Then, one of the states is selected by sampling a state according to the re-weighted probabilities.

As stated in Section 3, we perform experiments with char- and words-RNNs. We keep the same size and number of layers for both networks, although they result in different effective lengths; for example, manifold states are needed to

F:9 | D:min7 G:9 | C:maj F:9 | C:maj |	F:9 F:9 F:9 F:9 D:min7 D:min7 G:9 G:9 C:maj C:maj F:9 F:9 C:maj C:maj C:maj C:maj

Table 1: An example of the text representations of chord progressions in score (left) and the training data (right). A 4-bar chord progression is generally written in the form on the left, where the positions of the chords loosely indicate the chord change timings. On the right, the text show how the score on the left is represented in the training data. Here, the chords for every quarter notes are explicitly written and bar indicators are removed.

be predicted to complete a chord in char-RNNs while each state correspond to a chord in word-RNNs.

The dataset, code and audio files are released on web.[2]

3 Case Study 1: Chord progressions

3.1 Representation

The goal of this experiment is to generate chord progressions by training an LSTM network on jazz chord progressions. Here, we do not use any musical interpretation of the chords such as binary vectors to represent pitch and chords (as in [5]) but completely rely on their text representations. Table 1 shows an example of a chord progression and the corresponding texts. The left is an example of a chord notation in The Realbook score, where the positions of chords are loosely related to the timings of chord changes. The score on the left is converted into the text on the right, which specifies every chord for each quarter note.

We used $2,486$ scores from The Realbooks and The Fakebooks as training data. Every score file was parsed from *band-in-a-box* format to *.xlab* format. Then they were transposed to the key of C while every blank quarter note was filled with its preceding chord as in the Table 1. Finally, we put `_START_` and `_END_` flags (any distinctive words can be used as flags) at the beginning and the end of each score.

Although the key was transposed to C, only 867 (out of 2,846) scores end with C:maj (30%), followed by 489 G:7 (17%), 186 C:maj6 (7%), 52 F:maj (2%), and 1,252 scores end with the others – 237 chords (46%). This is because the The Realbook chord progressions usually end with chords for a *turn-around* to make the progressions natural to repeat the score.

[2] `https://github.com/keunwoochoi/lstm_real_book`
`https://github.com/keunwoochoi/LSTMetallica`
`https://soundcloud.com/kchoi-research/sets/lstm-realbook-1-5`
`https://soundcloud.com/kchoi-research/sets/lstmetallica-drums`

There were 1,259 unique chords in the training dataset. In other words, the vocabulary size of word-RNN was 1,259. However there were only 39 characters in total, which significantly reduced the computation of char-RNN. The total numbers of chords (words) and characters were 539,609 and 3,531,261, respectively.

3.2 Results

We set the system to output a chord progression for every diversity parameter α after every iteration. In this paper, we present four results from each networks (char-RNNs and word-RNNs), part of which are reported in the Table 2. For simplicity, we added bar symbols | and removed repeating chords in the same bar, e.g. | C:7 C:7 C:7 C:7 | reduced to | C:7 | and | C:7 C:7 E:min E:min | reduced to | C:7 E:min |.

First, both char-RNN and word-RNN showed well-structured results. They learned the local structures of chords and bars after sufficient number of iterations. In the result, the majority of chords continued for multiples of four, implying a single chord for within a bar. They also learned the local relationships between flags and chord. After one iteration, the flags are not placed properly as in the table 2 (a), where `_END_` is not followed by `_START_` but repeats itself.

i	α	**Chord progressions**
1	0.8	C:maj | G:7 | ... | G:7 | C:maj
1	1.2	A#:maj | A:7 | A:7 D:min7 D:min7 D:min7 | D:hdim C:hdim |C:hdim | C:hdim G:9 G:9 D:min7 | D:min7 D#:dim
23	0.8	C:7 F:maj|F:min|C:maj...C:maj|G:7| C:maj
23	1.2	C:7/5 C:7 | F:maj6 F#:dim | C:6(9) | C:6(9) | C:6(9) C:6(9) C:6(9) C:maj | E:7(b9) | A:min(6,9) A#:min(6,9) | A#:min(6,9) |... D:min | G:9 C:maj | ... G:7 `_END_` `_START_` C:maj

(a)

1	0.5	C:maj | G:7 | ... | G:7 | C:maj6
1	1.2	...C:maj `_END_` ... `_START_` `_START_` C#:maj A#:min A:sus4/5 C:maj/3 | F:min7 A:min7 D:min7 D:min7... `_START_`
8	0.5	C:maj A:min | D:min7 G:7(b9) | C:maj | A:min7 | D:9 | D:9 | D:7 | D:min7 | G:7 | C:maj | C:7 | F:maj | F:min | C:maj
8	1.2	C:Maj | G:min7 | F:maj | D:min7 D:min7 D:min7/4 G:sus4(b7) | G:min9 G:min9 G:min9 F#:(1,3,b5,b7,9,13) | C:6(9) G:sus4(b7,9) ... ... C:min `_END_` `_START_` C:maj

(b)

Table 2: Chord progressions generated by char-RNN (a) and word-RNN (b). Bar symbols (|) are inserted for readability and repeated chords in each bar are omitted.

As training continues, the flags start to appear in a sequence of `_END_ _START_` C:maj as in the training texts. The last chords of the score, i.e., the chord before `_END_` are not always same as the first chord (C), which is also natural as they vary in the training file.

Second, after sufficient training, both results showed chord progressions that lie in Jazz grammar. Examples are II-V-I progressions (D:min7- G:9 - C:maj), passing chords (A:dim - Ab:dim - G:min7), modal interchange chords (C:min6, Db:maj) and substitutions (B:7 as a tritone subdominant of F:7) in char-RNN; modal interchanges (G:min7), circle of fifths (Eb:sus - Gb:maj6 - B:maj7), and descending bass (C:maj6,9 - B:dim - A:min7 - Ab:7) in word-RNN. The authors noticed a subtle difference between the results from the two approaches. The results from word-RNN are more conventional progressions than those of char-RNN. However, it cannot be the fundamental difference of the two approaches. Instead, it may be caused by the difference of effective lengths between char- and word-RNNs layers - they have the same length of state sequences, but it results in a longer chord sequence in the word-RNN as mentioned in Section 2.2. In other words, the short memory of char-RNN may result in predictions that seem to be less constrained and stereotyped.

4 Case 2. Drum Tracks

4.1 Representation

There are issues when applying LSTM networks to drum tracks including finding a way to create and effective text representation. Both chord progressions and drum tracks are sequences of simultaneous events (pitches and drum components). However, drum tracks do not have a meaningful and compressive representation such as chord and it necessitate an encoding strategy of the track into text. We also need a finer time resolution as generally there are more than four events in a bar.

To encode simultaneous events in a track into texts, we used a binary representation of *pitches*, i.e., components of drums - kick, snare, hi-hats, cymbals, and tom-toms. For example, `100000000` and `010000000` represent kick and snare, respectively, and a simultaneous playing of kick and snare can be represented by `110000000`.

For efficient representation and learning, only nine components were allowed; kick, snare, open hi-hats, closed hi-hats, three tom-toms, crash cymbal, and ride cymbal.[3] We limited the number of events in a bar to 16 by quantising the drum track by 16th-note.

In the experiment, we first loaded 60 midi files of drum tracks of *Metallica* and quantised them. Then they were encoded into the above described binary

[3] Some of the components in the texts also represent other similar components, e.g. a closed hi-hats in the texts can mean either closed hi-hats or pedalled hi-hats in the original midi file.

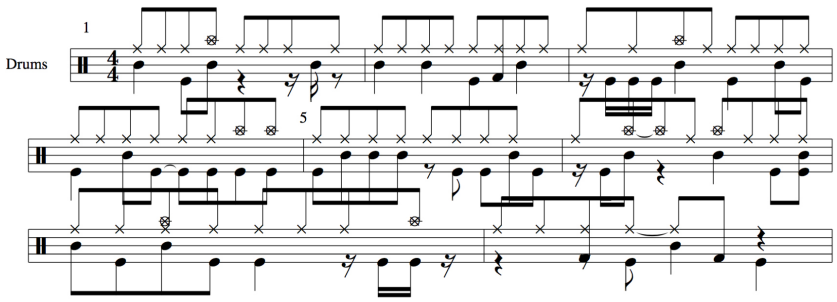

Fig. 1: A score of a generated drum track.

representation. We also added a flag `_BAR_` as an annotation of the bar segments in order to check if the networks learns the local structure.

There can be theoretically $2^9 = 512$ words, but there are supposedly much fewer words because the combinations of drum components that are played simultaneously are limited. The size of the word vocabulary in the training file is 119 and the file consists of 2,141,692 words in total.

4.2 Results

Char-RNNs turned out to fail to learn the drum tracks and output arbitrary 0's and 1's without any structures (the results have no spaces or `_BAR_` flags). The length of network may be too short to learn the long-term relationship between characters. In char-RNNs, representing a single bar requires 16 events$\times$10 characters=160 time steps. Encoding music sequences with only two characters - 0 and 1 (+space to for segmentation) - is an extreme approach for char-RNNs. In this paper, we therefore only report the result of word-RNNs.

Figure 1 shows one example of our results - a part of the generated track with $\alpha = 1.0$ after 25 iterations.[4] It consists of reasonable rock drum patterns - 8-beat hi-hats, combinations of kick and snare, and occasional crash cymbals and tom-toms. Although there are occasional kick/snare/tom-toms notes on back beats (of sixteen notes), hi-hats remain consistent, playing on 4-beat and 8-beat pattern, which is very common for instance in drum tracks of Metallica.[5]

Controlling α provides a way to tune the technical virtuosity of the track. Since large α increases the probabilities of occasional events, large α (=1.5) results in tracks with many fill-ins with tom-toms and a crash cymbal. On the other hands, when $\alpha < 1$, the track almost never contains anything but kick, snare, and hi-hats. As a result, it is possible to use a combination of small and large α in a drum track generator that is guided by user, who specifies where to add fill-ins.

[4] The score uses the percussion clef where $\times$ refers to hi-hats, notes on middle and bottom lines refers to snare and kick, respectively.

[5] `https://soundcloud.com/kchoi-research/00-24-100-bonus-for-score`, The score in the figure starts from 34-second.

5 Conclusion

We introduced an algorithm of text-based LSTM networks for automatic composition and reported results for generating chord progressions and rock drum tracks. Word-RNNs showed good results in both cases while char-RNNs only successfully learned chord progressions. The experiments show LSTM provides a way to learn the sequence of musical events even when the data is given as text. With the diversity parameter, the proposed algorithm can be used as a tool that helps human composers. In the future, a more complex network with the capability of learning interactions within music (instruments, melody/lyrics) will be examined for a more complete automatic composition algorithm.

References

1. Allan, M., Williams, C.K.: Harmonising chorales by probabilistic inference. Advances in neural information processing systems 17, 25–32 (2005)
2. Chollet, F.: Keras: Deep learning library for theano and tensorflow. https://github.com/fchollet/keras (2015)
3. De Mantaras, R.L., Arcos, J.L.: Ai and music: From composition to expressive performance. AI magazine 23(3), 43 (2002)
4. Eck, D., Schmidhuber, J.: A first look at music composition using lstm recurrent neural networks 103 (2002)
5. Franklin, J.A.: Recurrent neural networks for music computation. INFORMS Journal on Computing 18(3), 321–338 (2006)
6. Hiller, L., Isaacson, L.M.: Experimental Music. Composition with an Electronic Computer. McGraw-Hill Book Company (1959)
7. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001)
8. Jordan, M.I.: Attractor dynamics and parallellism in a connectionist sequential machine. Lawrence Erlbaum Associates (1986)
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR abs/1412.6980 (2014), http://arxiv.org/abs/1412.6980
10. Kleedorfer, F., Knees, P., Pohle, T.: Oh oh oh whoah! towards automatic topic detection in song lyrics. In: ISMIR. pp. 287–292 (2008)
11. Lambert, A.J., Weyde, T., Armstrong, N.: Perceiving and predicting expressive rhythm with recurrent neural networks (2015)
12. Lewis, J.: Algorithms for music composition by neural nets: Improved cbr paradigms. In: International Computer Music Conference (1989)
13. Mozer, M.C.: Neural network music composition by prediction. Connection Science 6(2-3), 247–280 (1994)
14. Simon, I., Morris, D., Basu, S.: Mysong: automatic accompaniment generation for vocal melodies. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 725–734. ACM (2008)
15. Sutskever, I., Martens, J., Hinton, G.E.: Generating text with recurrent neural networks. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11). pp. 1017–1024 (2011)
16. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)