ATIAM 2017 - ML Project
Generation of lead sheets and inference in jazz through recurrent neural networks

Tristan Carsault, Philippe Esling*
Institut de Recherche et Coordination Acoustique Musique (IRCAM)
UPMC - CNRS UMR 9912 - 1, Place Igor Stravinsky, F-75004 Paris
{tristan.carsault, esling}@ircam.fr

Abstract

This project aims to generate lead sheets of jazz music, represented as coherent chord label sequences with the help of probabilistic models. The motivation for this approach comes from a recent article on text-based LSTM networks for automatic music composition Choi et al. [2016]. In this paper, the authors use a recurrent neural network (RNN-LSTM) to generate symbolic chord sequences. They focus on two different approaches named *word-RNN* and *char-RNN*. These two variants use the same model architecture but rely on different learning methods. In this project, we will improve the word-RNN approach by injecting music theory knowledge through the learning method in order to be able to perform accurate prediction of chord sequence and jazz melody generation. Ultimately, this project could be used to perform automatic accompaniment and improvisation.

1 Introduction

Symbolic music generation is a field that has been widely explored through Hidden Markov Models (HMM). For instance, Paiement, Bengio and Eck published a probabilistic models for melodic prediction using chord instantiation Paiement et al. [2005]. Nevertheless, HMMs are bounded by their order of complexity. On the opposite, deep learning techniques are capable of computing highly non-linear functions and can extract information from complex data. Therefore, most recent works in the symbolic music generation field are now based on Neural Networks (NN), and particularly on Recurrent Neural Networks (RNN).

One of the largest issue in NNs is its incapacity to correctly store temporal information. However, in our setup, we need to discover the underlying structure of symbolic musics. Therefore, we need to design our model with an understanding of time. Recurrent Neural Networks (RNN) show promising results in many learning fields Graves [2013] including musical structure generation. To define recurrent networks, we usually rely on loops inside the network, where the output of a neuron is directly linked to itself for the next sequence element. The training of RNNs is performed by Backpropagation Through Time (BPTT). This algorithm works by fixing an amount of time steps, unrolling the RNN in order to obtain a classic NN, and training it with a standard back-propagation.

Nevertheless, standard RNNs tend to be unable to process long-term relationships correctly and quickly forget information across time steps. Hence, gated memory mechanisms solved this problem by adding trainable gates to standard RNNs that allow to select, stock and output the most important information for the task at end. The two most used gated RNN structures are the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU).

*https://esling.github.io/atiam-ml-project

2 Toy dataset

Your first assignment, in order to gain an understanding of the problem, is to create toy datasets that will be used to test different models. As we work on specific data which are chord label progressions, we propose to create a toy dataset that follows the behavior of chord progressions. Thus, we must focus on two main aspects : relations between labels and repetition of labels. This toy dataset will be used to analyze the models in simplified setups in order to understand their behavior.

Exercice 1 - Designing a toy dataset In the field of symbolic music generation and chord inference, some typical constructions exist, while other remain under-studied. Therefore, defining a toy dataset is an open question and your proposed solution will be evaluated on the design and justification of your choices. Hence, we highly recommend that you use both your personal knowledge, music theory analysis books and eventually to propose some innovative approaches to the question.

In this case, the following questions must be answered separately (and therefore, produce two sets of scripts and data), for *chord progressions* and *melodic progressions*

1. Describe some seminal and prototypical examples that should minimally be understood by any system trying to tackle the question of musical inference
2. Describe the factors of variations that could exist in such datasets
3. Describe situations in which these sequences require multiple scales of time
4. Code different *procedural functions* that could generate large sets of examples following all of your observations in the previous questions
5. Generate your sets of data (textual or midi) in a structured way
6. Explain how these sets can be labeled and analyzed

3 Models and expected work

In order to understand the models and to be able to code these by yourself, we will start by understanding the basic RNN and LSTMs, and perform their implementation by relying on the *Keras* framework for Python. Tutorials are available at `http://keras.io`

Exercice 2 - Learning Keras and understanding RNN/LSTM As you might see in the Keras tutorials, this framework propose some pre-developed layer for both LSTMs and RNNs. However, this might not help you to understand their inner functioning. Therefore, we will try to better understand those architectures and to code their simpler version by ourselves. Here, we will rely on the tutorial `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

1. Based on the description of the tutorial, try to develop your own RNN and own LSTM layer (using Keras layer definition `https://keras.io/layers/writing-your-own-keras-layers/`). You can also use other sources of information.
2. Compare your models to the RNN and LSTM already provided in Keras
3. Train all models on your toy datasets and compare performance
4. Analyze and explain the behavior of the models for different properties
5. Find interesting visualizations to understand this behavior

Exercice 3 - Understanding the reference paper After this stage, all our models will be trained on the *real book dataset* Choi et al. [2016] composed of 2847 tracks taken from the real book [2] and by using the model described in the reference paper. The starting code and dataset are available at this link : `https://github.com/keunwoochoi/lstm_real_book`.

You can also find the article `https://arxiv.org/pdf/1604.05358v1.pdf` and blog post related to this `https://keunwoochoi.wordpress.com/2016/02/19/lstm-realbook/`.

[2] The real book dataset is a compilation of jazz classics that has been firstly collected by the students of the Berklee College of Music during the seventies. As of today we count a lot of existing books (e.g Realbook (1, 2, 3), New Realbook (1, 2, 3), the Fakebook, the real Latin book)

```
<chord>      ::=    <pitchname> ":" <shorthand> ["("<ilist>")"]["/"<interval>]
                  | <pitchname> ":" "("<ilist>")" ["/"<interval>]
                  | <pitchname> ["/"<interval>]
                  | "N"

<pitchname>  ::=  <natural> | <pitchname> <modifier>

<natural>    ::=  "A" | "B" | "C" | "D" | "E" | "F" | "G"

<modifier>   ::=  "b" | "#"

<ilist>      ::=  ["*"] <interval> ["," <ilist>]

<interval>   ::=  <degree> | <modifier> <interval>

<degree>     ::=  <digit> | <digit> <degree> | <degree> "0"

<digit>      ::=  "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<shorthand>  ::=   "maj" | "min" | "dim" | "aug" | "maj7" | "min7" | "7"
                  | "dim7" | "hdim7" | "minmaj7" | "maj6" | "min6" | "9"
                  | "maj9" | "min9" | "sus2" | "sus4"
```

Figure 1: Syntax of Chord Notation in the Realbook dataset, image taken from Harte [2010]

1. Based on the paper, try to re-implement the model.
2. Train all models on both the toy datasets and the real book set.
3. What happens if you train the paper model on the real book set and then just fine tune it on the toy datasets ?

3.1 Alphabets and chord distances

The original *word-RNN* described in the paper uses a vocabulary of 1259 labels. This huge amount of labels comes from the complexity of the original chord syntax described in Figure 1. Nevertheless, we are more interested here in the functional aspect of the chord labels than in the precise description. Thus, we propose first to apply a clustering of the elements from the initial alphabet A_0 to the three following alphabets (the mapping will be provided) :

A_1 : Major and minor: N (which means no chord), maj, min;

A_2 : Major and minor, seventh, diminished: N, maj, min, maj7, min7, 7, dim;

A_3 : Major and minor, seventh, diminished, augmented: N, maj, min, maj7, min7, 7, dim, aug;

Exercice 4 - Evaluating alphabet reductions The alphabet reduction could be done at different step of the prediction (before/after the NN). For instance, we apply a first reduction $A_0 \rightarrow A_3$ before the NN and a second $A_3 \rightarrow A_2$ on the outputs. Do we obtain the same prediction results than with the reduction $A_0 \rightarrow A_2$ directly on the inputs ?

1. Setup and justify different types of reductions
2. Compare their effects and results

Exercice 5 - Improving the cost function In the paper, the cost function is a categorical crossentropy. We want to upgrade this cost function with music theory consideration. Thus, we will modify it in order to train our model with other chord distances

1. Implement the Tonnetz distance
2. Implement the Euclidian distance between semitone vectors (The code for the chord to vector transformation will be provided)
3. (Bonus) Another distance of your inspiration based on the music/perceptive/cognitive theory
4. Compare the results of various models (also with respect to toy datasets)

The main modifications will occur on the loss function at line 33 of the main project file `https://github.com/keunwoochoi/lstm_real_book/blob/master/main_lstm_realbook.py`. We give useful links to understand loss functions in the Keras environment `https://keras.io/losses` and by using its back-end (Tensorflow) `https://github.com/fchollet/keras/blob/master/keras/losses.py`.

We will then combine the different alphabets with the different distances and compare our results thanks to the prediction score on a test dataset.

3.2 Connectionist Temporal Classification

In this second part, we focus on the coherence of the generated sequences. Indeed, chord sequences on which our model is trained are often composed with successive similar chords. Thus, the network naturally predicts the new element as the last chord of the seed sequence. In order to minimize this effect, Keunwoo Choi et al. introduce a diversity parameter that penalize redundancies in the generate sequence by applying a basic sampling on the duplicated elements in the generated sequence. Our wish is then to introduce a new learning method based on the CTC Graves et al. [2006]. This method allows to time-stretch the generated sequence and to select only useful information.

Exercice 6 - Relying on the CTC cost

1. Change the cost function so that it uses the CTC approach described in `ftp://ftp.idsia.ch/pub/juergen/icml2006.pdf` (You can also use external sources of information)
2. Compare your results to the diversity parameter method.

References

Keunwoo Choi, George Fazekas, and Mark Sandler. Text-based lstm networks for automatic music composition. *arXiv preprint arXiv:1604.05358*, 2016.

Jean-François Paiement, Douglas Eck, and Samy Bengio. A probabilistic model for chord progressions. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR)*, number EPFL-CONF-83178, 2005.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Christopher Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, 2010.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.