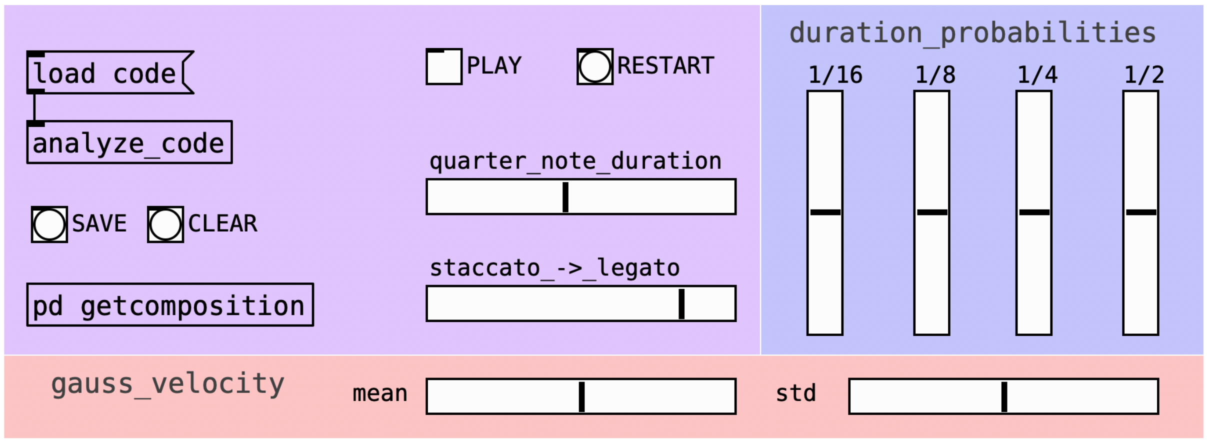
Markov chain music

Davi Bedoya & Christian Steinmetz
12.02.2020

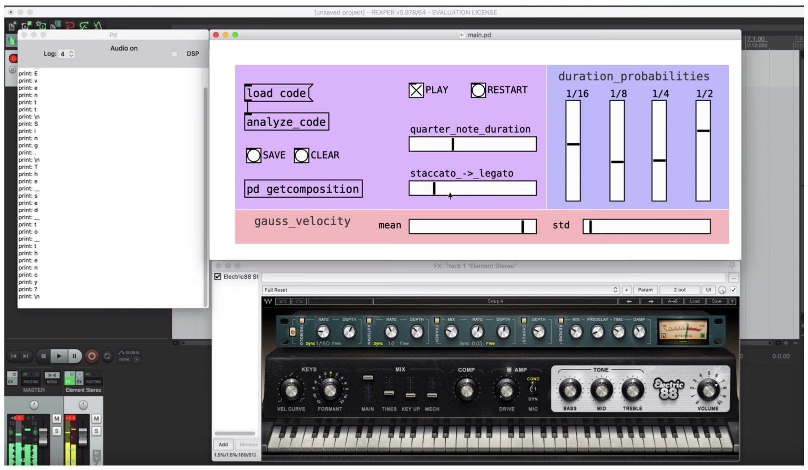
We use a second-order markov model on tokenized poems (at the character level) to write and sonify new poems.



Usage

To start composing you must first load a properly formatted file which contains a list of tokenized characters take from a text corpus. We provide two pre-made files: `lyrics_code_ch.txt` (taken from song pop song lyrics) and `poems_code_ch.txt` (taken from Shakespeare poems). We also provide a Jupyter notebook (`convert_ch.ipynb`) that will allow you to use any text file corpus to create properly formatted input files.

After you press **load code** and select a file, it will be analyzed and the markov model will be initialized. To begin creating music all you need is to press the **PLAY** toggle and ensure that you have an external MIDI instrument listening for notes.



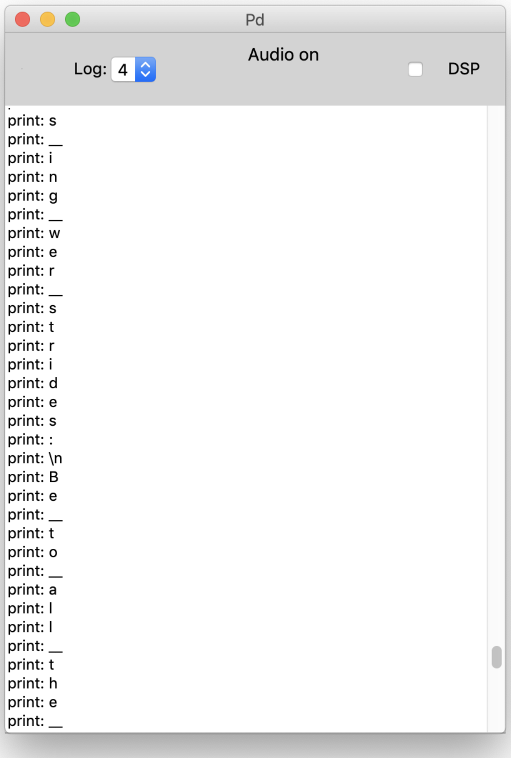
See an example composition here

Controls

PLAY - As mentioned above this toggle starts and stops the generation of new characters and MIDI notes.
RESTART - This will restart the composition process by re-seeding the generation to the start of the supplied corpus.
quarter_note_duration - This slider lets you control how fast the overall tempo of the composition is.
duration_probabilities - With these four sliders you can control how likely each type of note is to be generated. This can help you dynamically change the rhythmic aspects.
staccato_->_legato - With this control you can change the relative duration of a note with respect to its note duration.
gauss_velocity - Here we provide two controls that let you adjust the dynamics of the performance. Changing the mean will adjust overall how loud or soft the notes are played, and increasing the std will result in more variation on each note.
SAVE - This can be used to save the current text of the composition to disk (currently `./composition.txt`).
CLEAR - This will clear the composition cache.

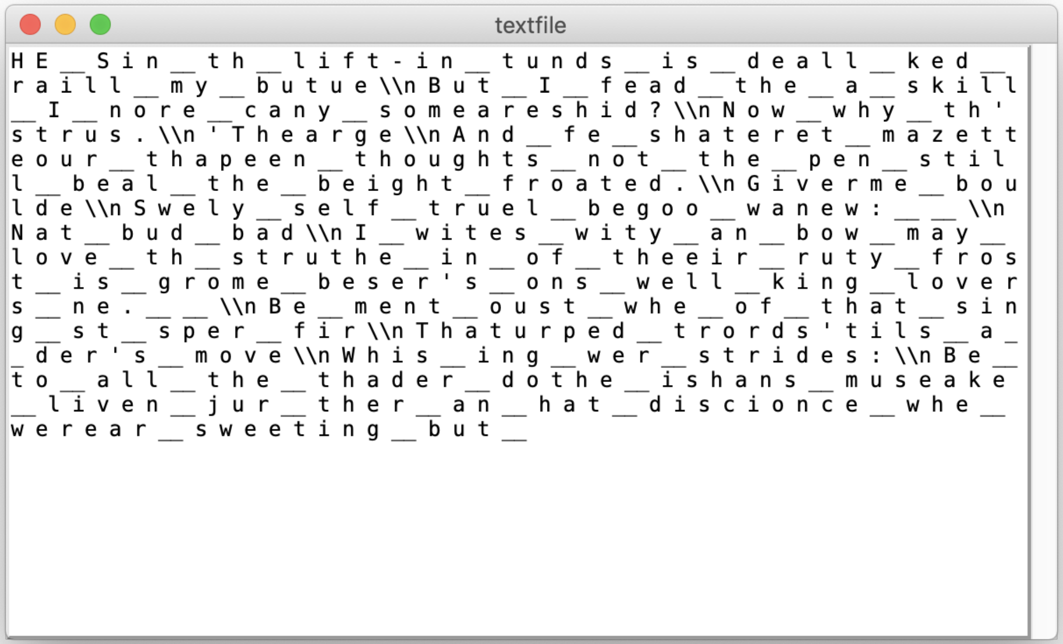
Features

One of the interesting features is that we are generating not only music but a poem that accompanies the music, which we then sonify in real-time. You can see the poem as it is being written in the terminal (as shown below). In this case we use `__` to specify a space between words and `\n` for newline.



```
print: s
print: __
print: i
print: n
print: g
print: __
print: w
print: e
print: r
print: __
print: s
print: t
print: r
print: i
print: d
print: e
print: s
print: :
print: \n
print: B
print: e
print: __
print: t
print: o
print: __
print: a
print: l
print: l
print: __
print: t
print: h
print: e
print: __
```

In addition, we write all the notes being generated to a textfile so you can save your composition later and read it. The notation here is a bit cluttered and a future room for improvement would be to have a scheme that allows for us to save this data out in a better way along with the tokenized version, so we could feed back in poems we generated ourselves to the model.



```
H E __ S i n __ t h __ l i f t - i n __ t u n d s __ i s __ d e a l l __ k e d __
r a i l l __ m y __ b u t u e \\n B u t __ I __ f e a d __ t h e __ a __ s k i l l
__ I __ n o r e __ c a n y __ s o m e a r e s h i d ? \\n N o w __ w h y __ t h '
s t r u s . \\n ' T h e a r g e \\n A n d __ f e __ s h a t e r e t __ m a z e t t
e o u r __ t h a p e e n __ t h o u g h t s __ n o t __ t h e __ p e n __ s t i l
l __ b e a l __ t h e __ b e i g h t __ f r o a t e d . \\n G i v e r m e __ b o u
l d e \\n S w e l y __ s e l f __ t r u e l __ b e g o o __ w a n e w : __ __ \\n
N a t __ b u d __ b a d \\n I __ w i t e s __ w i t y __ a n __ b o w __ m a y __
l o v e __ t h __ s t r u t h e __ i n __ o f __ t h e e i r __ r u t y __ f r o s
t __ i s __ g r o m e __ b e s e r ' s __ o n s __ w e l l __ k i n g __ l o v e r
s __ n e . __ __ \\n B e __ m e n t __ o u s t __ w h e __ o f __ t h a t __ s i n
g __ s t __ s p e r __ f i r \\n T h a t u r p e d __ t r o r d s ' t i l s __ a _
_ d e r ' s __ m o v e \\n W h i s __ i n g __ w e r __ s t r i d e s : \\n B e __
t o __ a l l __ t h e __ t h a d e r __ d o t h e __ i s h a n s __ m u s e a k e
__ l i v e n __ j u r __ t h e r __ a n __ h a t __ d i s c i o n c e __ w h e __
w e r e a r __ s w e e t i n g __ b u t __
```

Collaboration

In this project, Davi developed the idea to use poems as a unique corpus for music generation. He also developed the Python code to allow us to tokenize different text corpora as input to our patch. Christian and Davi worked together to further refine the overall concept of operation. Davi's work also focused on the design of a system to have weighted probabilities of the different note durations. Christian worked to develop additional ways to impart randomness in the composition, as well as the methods to save and visualize the compositions in real-time. Christian also created the final demonstration videos. This was a fruitful collaboration we the combination of our ideas led us to an interesting result that had many features, and overall a nice interface for not only generating poems and sonifying them, but also for interacting with the composition process in real-time.