Magenta RNN

In this notebook, we will be generating three basic melodies using Magenta and it's three models.

In [45]:

```
import math
import os
import time
import warnings
warnings.filterwarnings('ignore')

import magenta.music as mm
from magenta.models.melody_rnn import melody_rnn_sequence_generator
from magenta.music import DEFAULT_QUARTERS_PER_MINUTE
from magenta.protobuf.generator_pb2 import GeneratorOptions
from magenta.protobuf.music_pb2 import NoteSequence
from visual_midi import Plotter

def generate(bundle_name: str,
             sequence_generator,
             generator_id: str,
             primer_filename: str = None,
             qpm: float = DEFAULT_QUARTERS_PER_MINUTE,
             total_length_steps: int = 64,
             temperature: float = 1.0,
             beam_size: int = 1,
             branch_factor: int = 1,
             steps_per_iteration: int = 1,
             show_plot: bool = False) -> NoteSequence:
    mm.notebook_utils.download_bundle(bundle_name, "bundles")
    bundle = mm.sequence_generator_bundle.read_bundle_file(os.path.join("bundles", bundle_name))
    generator_map = sequence_generator.get_generator_map()
    generator = generator_map[generator_id](checkpoint=None, bundle=bundle)
    generator.initialize()
    if primer_filename:
        primer_sequence = mm.midi_io.midi_file_to_note_sequence(
          os.path.join("simplemidi", primer_filename))
    else:
        primer_sequence = NoteSequence()
    if primer_sequence.tempos:
        if len(primer_sequence.tempos) > 1:
          raise Exception("No support for multiple tempos")
        qpm = primer_sequence.tempos[0].qpm
    # Calculates the seconds per 1 step, which changes depending on the QPM
    # value (steps per quarter in generators are mostly 4)
    seconds_per_step = 60.0 / qpm / getattr(generator, "steps_per_quarter", 4)

    # Calculates the primer sequence length in steps and time by taking the
    # total time (which is the end of the last note) and finding the next step
    # start time.
    primer_sequence_length_steps = math.ceil(primer_sequence.total_time
                                             / seconds_per_step)
    primer_sequence_length_time = (primer_sequence_length_steps
                                   * seconds_per_step)
    primer_end_adjust = (0.00001 if primer_sequence_length_time > 0 else 0)
    primer_start_time = 0
    primer_end_time = (primer_start_time
                       + primer_sequence_length_time
                       - primer_end_adjust)
    generation_length_steps = total_length_steps - primer_sequence_length_steps
    if generation_length_steps <= 0:
        raise Exception("Total length in steps too small "
                        + "(" + str(total_length_steps) + ")"
                        + ", needs to be at least one bar bigger than primer "
                        + "(" + str(primer_sequence_length_steps) + ")")
    generation_length_time = generation_length_steps * seconds_per_step
    generation_start_time = primer_end_time
    generation_end_time = (generation_start_time
                           + generation_length_time
                           + primer_end_adjust)
    # Showtime
    print("Primer time: ["
          + str(primer_start_time) + ", "
          + str(primer_end_time) + "]")
    print("Generation time: ["
          + str(generation_start_time) + ", "
          + str(generation_end_time) + "]")
    generator_options = GeneratorOptions()
    generator_options.args['temperature'].float_value = temperature
    generator_options.args['beam_size'].int_value = beam_size
    generator_options.args['branch_factor'].int_value = branch_factor
    generator_options.args['steps_per_iteration'].int_value = (
        steps_per_iteration)
    generator_options.generate_sections.add(
        start_time=generation_start_time,
        end_time=generation_end_time)
    sequence = generator.generate(primer_sequence, generator_options)

    date_and_time = time.strftime('%Y-%m-%d_%H%M%S')
    generator_name = str(generator.__class__).split(".")[2]
    midi_filename = "%s_%s_%s.mid" % (generator_name, generator_id,
                                      date_and_time)
    midi_path = os.path.join("output", midi_filename)
    mm.midi_io.note_sequence_to_midi_file(sequence, midi_path)
    print("Generated midi file: " + str(os.path.abspath(midi_path)))

    # Writes the resulting plot file to the output directory
    date_and_time = time.strftime('%Y-%m-%d_%H%M%S')
    generator_name = str(generator.__class__).split(".")[2]
    plot_filename = "%s_%s_%s.html" % (generator_name, generator_id,
                                       date_and_time)
    plot_path = os.path.join("output", plot_filename)
    pretty_midi = mm.midi_io.note_sequence_to_pretty_midi(sequence)
    plotter = Plotter()
    if show_plot:
        plotter.show(pretty_midi, plot_path)
    else:
        plotter.save(pretty_midi, plot_path)
    print("Generated plot file: " + str(os.path.abspath(plot_path)))

    return sequence
```

In [46]:

```
sequence = generate(
    "basic_rnn.mag",
    melody_rnn_sequence_generator,
    "basic_rnn",
    primer_filename="twinkle.mid",
    total_length_steps=64,
    temperature=0.9,
    show_plot=True)
```

```
WARNING:tensorflow:The saved meta_graph is possibly from an older release:
'model_variables' collection should be of type 'byte_list', but instead is of type 'node_list'.
INFO:tensorflow:Restoring parameters from
/var/folders/dm/3kslprps6b736vz2bgdqpwx00000gn/T/tmpc0hbdxeq/model.ckpt
Primer time: [0, 0.87499]
Generation time: [0.87499, 8.0]
INFO:tensorflow:Beam search yields sequence with log-likelihood: -41.987698
Generated midi file: /Users/brian/422magenta/output/melody_rnn_basic_rnn_2020-05-07_173401.mid
Generated plot file: /Users/brian/422magenta/output/melody_rnn_basic_rnn_2020-05-07_173401.html
```

In [47]:

```
sequence = generate(
    "lookback_rnn.mag",
    melody_rnn_sequence_generator,
    "lookback_rnn",
    primer_filename="twinkle.mid",
    total_length_steps=64,
    temperature=0.9,
    show_plot=True)
```

```
WARNING:tensorflow:The saved meta_graph is possibly from an older release:
'model_variables' collection should be of type 'byte_list', but instead is of type 'node_list'.
INFO:tensorflow:Restoring parameters from
/var/folders/dm/3kslprps6b736vz2bgdqpwx00000gn/T/tmpmfqdpy5c/model.ckpt
Primer time: [0, 0.87499]
Generation time: [0.87499, 8.0]
INFO:tensorflow:Beam search yields sequence with log-likelihood: -81.714355
Generated midi file: /Users/brian/422magenta/output/melody_rnn_lookback_rnn_2020-05-07_173414.mid
Generated plot file: /Users/brian/422magenta/output/melody_rnn_lookback_rnn_2020-05-07_173414.html
```

In [48]:

```
sequence = generate(
    "attention_rnn.mag",
    melody_rnn_sequence_generator,
    "attention_rnn",
    primer_filename="twinkle.mid",
    total_length_steps=64,
    temperature=0.9,
    show_plot=True)
```

```
WARNING:tensorflow:The saved meta_graph is possibly from an older release:
'model_variables' collection should be of type 'byte_list', but instead is of type 'node_list'.
INFO:tensorflow:Restoring parameters from
/var/folders/dm/3kslprps6b736vz2bgdqpwx00000gn/T/tmpxukj2qp4/model.ckpt
Primer time: [0, 0.87499]
Generation time: [0.87499, 8.0]
INFO:tensorflow:Beam search yields sequence with log-likelihood: -37.159786
Generated midi file: /Users/brian/422magenta/output/melody_rnn_attention_rnn_2020-05-07_173421.mid
Generated plot file: /Users/brian/422magenta/output/melody_rnn_attention_rnn_2020-05-
07_173421.html
```

In [ ]:

In [ ]: