# Musical Ngrams

2018-01-30 BY GENE

What are the most repeated phrases of musical compositions?  Naturally I wrote a program to tell me!

Here is the first part that declares the preamble and modules to use:

```
#!/usr/bin/env perl

# Play the top repeated note phrases of a MIDI file.

use strict;
use warnings;

use MIDIUtil;
use MIDI;
use Lingua::EN::Ngram;
use List::Util qw( shuffle );
```

This uses my simple MIDIUtil module to set things up.

Next up, we take input from the command-line user:

```
my $file = shift || die "Usage: perl $0 /some/file.mid size max bpm randomi
my $size = shift || 2;   # ngram size
my $max  = shift || 40;  # -1 for all >1 records
my $bpm  = shift || 100; # Beats per minute
my $ranp = shift || 0;   # Random patch instead of all piano
my $shuf = shift || 0;   # Shuffle phrases
my @durations = @ARGV ? @ARGV : qw( tqn qn );
```

(Sadly, the ngram rhythms are not preserved – only the pitches.  Instead, we give the program a set of durations to choose from at random.)

This is followed by some variables that will be used:

```
# General MIDI patches that are audible and aren't horrible
my @patches = qw(
    0 1 2 4 5 7 8 9
    13 16 21 24 25 26
    32 34 35 40 42 60
    68 69 70 71 72 73
    74 79
);

my $opus = MIDI::Opus->new( { from_file => $file } );

# Bucket of note phrases
my %notes;

# Counter for the tracks seen
my $i = 0;
```

Now for the first procedure of the program: Turn the MIDI note information into strings of phrases (e.g. “71 69 71 55” -> “hb gj hb ff”) and then into ngram chunks:

```
# Handle each track...
for my $t ( $opus->tracks ) {
    # Collect the note events for each track but channel 9 (percussion)
    my @events = grep { $_->[0] eq 'note_on' && $_->[2] != 9 && $_->[4] !=

    my $track_channel = $events[0][2];

    # Skip if there are no events and no channel
    next unless @events && defined $track_channel;
    $i++;
    print "$t $i\n";

    # Declare the notes to inspect
    my $text = '';

    # Accumulate the notes
    for my $event ( @events ) {
        ( my $num = $event->[3] ) =~ tr/0-9/a-j/;
        $text .= "$num ";
    }

    # Parse the note text into ngrams
    my $ngram  = Lingua::EN::Ngram->new( text => $text );
    my $phrase = $ngram->ngram($size);

    # Counter for the ngrams seen
    my $j = 0;

    # Display the ngrams in order of their repetition amount
    for my $p ( sort { $phrase->{$b} <=> $phrase->{$a} } keys %$phrase ) {
        next if $phrase->{$p} == 1; # Skip single occurance phrases

        $j++;

        # End if we are past the maximum
        last if $max > 0 && $j > $max;

        ( my $num = $p ) =~ tr/a-j/0-9/;

        printf "\t%d.\t%d\t%s\n", $j, $phrase->{$p}, $num;
        push @{ $notes{$track_channel} }, $num;
    }
}
```

Next up is to actually construct a MIDI file from these phrases:

```
die "\n* Can't handle songs with more than 16 tracks.\n"
    if keys(%notes) > 16;

my $score = MIDIUtil::setup_midi( bpm => $bpm );

my @phrases;
my $channel = 0;

# Generate a function for the notes of each track
for my $track ( keys %notes ) {
    my @all;

    my @track_notes = $shuf ? shuffle @{ $notes{$track} } : @{ $notes{$trac

    # Shuffle the phrases and add the notes to a bucket
    for my $phrase ( @track_notes ) {
        my @phrase = split /\s/, $phrase;
        push @all, @phrase;
    }

    # Create a function that adds our bucket of notes to the score
    my $func = sub {
        $channel++;

        my $patch = $ranp ? random_patch() : 0;

        MIDIUtil::set_chan_patch( $score, $channel, $patch);

        for my $note ( @all ) {
            my $duration = $durations[ int rand @durations ];
            $score->n( $duration, $note );
        }
    };

    push @phrases, $func;
}

$score->synch(@phrases);

$score->write_score( "$0.mid" );

sub random_patch {
    return $patches[ int rand @patches ];
}
```

I ran Bach's Jesu Joy of Man's Desiring (bach_jesu_joy_with_piano) through this program and generated this file: note-ngram-play-JESU.

Here is the text output showing the index, the first 20 repetitions and the 4 note phrase itself (in MIDI note number notation):

```
MIDI::Track=HASH(0x7fc9640a4570) 1
        1.      10      71 69 71 72
        2.      8       72 74 71 69
        3.      8       71 72 74 71
        4.      6       74 71 69 71
        5.      6       71 72 71 69
        6.      6       71 72 74 74
        7.      6       74 72 71 69
        8.      6       72 74 74 72
        9.      6       69 71 72 74
        10.     6       69 71 72 71
        11.     6       74 74 72 71
        12.     6       72 71 69 67
        13.     4       72 71 69 71
        14.     4       72 72 71 72
        15.     4       72 71 72 74
        16.     3       67 71 72 74
        17.     3       69 67 71 72
        18.     3       71 69 67 71
        19.     2       74 76 77 74
        20.     2       72 71 69 69
MIDI::Track=HASH(0x7fc9640a4f00) 2
        1.      6       67 67 59 60
        2.      6       67 67 67 59
        3.      6       59 60 62 62
        4.      6       67 59 60 62
        5.      6       60 62 62 55
        6.      4       66 67 62 66
        7.      4       67 69 66 67
        8.      4       67 67 69 66
        9.      4       69 66 67 62
        10.     3       62 55 67 67
        11.     3       55 67 67 69
        12.     3       62 62 55 67
        13.     2       62 64 60 62
        14.     2       72 71 60 60
        15.     2       62 66 66 67
        16.     2       71 60 60 67
        17.     2       55 66 68 69
        18.     2       66 68 69 69
        19.     2       60 62 62 67
        20.     2       67 62 66 67
MIDI::Track=HASH(0x7fc9640e8148) 3
        1.      17      71 69 71 55
        2.      16      52 74 71 67
        3.      16      47 79 78 79
        4.      16      74 47 79 78
        5.      16      76 74 74 47
        6.      16      79 78 79 52
        7.      16      79 52 74 71
        8.      16      74 74 47 79
        9.      16      78 79 52 74
        10.     15      72 52 76 74
        11.     15      74 72 72 52
        12.     15      55 67 69 71
        13.     15      72 72 52 76
        14.     15      71 55 67 69
        15.     15      52 76 74 74
        16.     13      67 62 50 67
        17.     13      69 71 64 48
        18.     13      71 64 48 74
        19.     13      69 71 55 67
        20.     13      69 67 62 50
```

And here is what it sounds like after re-assigning the piano patches:

00:00 00:00

Ok. How about some Beethoven?  Here is the Moonlight Sonata in 8 note phrases with different durations given: note-ngram-play-MOONLIGHT

And here is what that sounds like when put through my DAW:

00:00 00:00

Here is Peter Gabriel's "Games Without Frontiers" in chunks of 4 note phrases, made with this command, and then re-orchestrated with my DAW:

```
perl -I. note-ngram-play ~/Music/peter_gabriel-games_without_frontiers.mid
```

00:00 00:00

How about “Big Time” from the same site? This *actually* sounds like old school Peter Gabriel when re-orchestrated with flutes and mallets.

00:00 00:00

~

**UPDATE**: A *vastly superior* update to this program that includes documentation and uses “weighted choice” to determine the phrases to play is ngram-play. And now *even more superior* module is now on CPAN. Woo! And here is a Web GUI app that I made to make this analysis easy: App-MIDI-Ngram

FILED UNDER: MUSIC, SOFTWARE
TAGGED WITH: NGRAMS, PERL

Epistemologist-at-large