CS 221: Progress Report

Mathieu Rolfo, Shalom Rottman-Yang, Nathan Tindall

13 November 2014

Rap Lyrics to Poetry

Updates from Project Proposal

In our original project proposal, we described our plan to utilize rap lyrics to predict the results of music reviews. This idea was inspired by existing data aggregation that has been performed on Rap Genius corpus, which made us optimistic that data from that source would be possible to harvest and manipulate. The content of lyrics can be measured on several metrics, including corpus frequency, *n*-grams, sentiment analysis, and higher level features, making this area of investigation an interesting and fruitful one. However, our original plan involved the scraping and aggregation of music reviews, which has turned out to be somewhat of a challenge. Additionally, while we may have been able to fine tune the features in order to increase accuracy, we realized that perfecting a classifier and predictor is somewhat less algorithmically interesting than exploring other ideas we had.

Problem Summary

Our project will take the rap corpus (input) and generate poetry (output) based on the corpus text. We will implement a generative search algorithm that conforms to the constraints of various poetic models, such as sonnets and or haiku. We generally define poetry as a set of strings of text with formal constraints. A lot of work has been done on generative algorithms, especially with respect to text; however, we believe that our project explores new territory in two respects: **(1)** the characteristics of the corpus are unusual in that it contains slang words, incomplete grammatical structures, and various lexical phenomena uncommon to regular prose; and, **(2)** the constraints of poetry are extensive with respect to (a) syllabic count and rhyme, (b) number of lines, and (c) prosody (metrical foot). **(1)** and **(2)** impose constraints that must be incorporated into our implementation, outside the normal generative text algorithm. These factors, in tandem, make the exploration space an exciting one.

Model

Our model is defined by a poetry object. The parameters of the poetry object are defined at runtime or specified by particular classes of poems (e.g. haiku, sonnet), and include the number of lines in the poem, the metrical foot of the poem, the rhyming scheme of the poem, and the number of syllables that are required on each line. We have aggregated corpus files of various artists' from RapGenius. Our algorithm will be trained on a corpus file of a given rapper, specified at runtime, in order to generate output in accordance with the constraints of the particular poetry model. Our poems are comprised of lines. Each line in our model will have syllabic constraints: with haiku, for example, the first line must have exactly five syllables. Our poem object will also have constraints between lines: with Italian Sonnets, for example, lines 0 and 3 must rhyme. Our problem is set up as a backtracking search problem, where search backtracks when the poetry model is violated. Our implementation will be depth first search: find a solution that satisfies all of the given constraints for each line and between lines.

Baseline

Our baseline is a probabilistic *n*-gram model that produces strings after being trained on an artist's corpus. The baseline lacks rhyming constraints and semantic checking; however, it is able to produce rhetoric that uses the artist's lexicon, relying on the *n*-gram model to produce semantically consistent prose. As a simplifying assumption, the baseline assumes that a line of poetry has eight words, as a stand-in for syllabic constraints. The first word of each line is based on the *n*-gram value of the last *n* words on the previous line. Additionally, the baseline assumes that poetry has 15 lines and a standard form (same number of syllables per line, etc.). See below excerpts of two examples of poems, trained on the Kanye West corpus:

(*n*=1)

```
tryin' to see is spiritual sand To the
lights and say about clothes Don't worry 'bout
to go figure Told some work that I
pray like playing And this is he says,
"who had her nickname Sorrow My iPhone need
```

```
(n=2)
say, lost my partner rock a mink coat
in the crew But them niggas creeping around
the world, if the songs of their game
and then I be, be the ones I
was clapping in the night off So we
```

Our analysis of the output generated by the *n*-gram model is that while the *1*-gram model produces phrases that are syntactically and semantically inconsistent, the *3*-gram model frequently just reproduces the lines verbatim. In the scope of the baseline, this may be a function of the size of the corpus alone, such that a *3*-gram model may succeed at generating novel output once rhyming and syllabic constraints are imposed, however, it could also be an indicator that *n*-gram features alone may not be sufficient in order to produce novel output, given the size of our corpus. This may prompt us to add additional syntactic feature indicators as we proceed in our developmental process.

Oracle

The oracle for this problem is asking a human to write poetry. This is something that humans can do with very good accuracy, creativity, and expression. The fact that our algorithm draws from the lexicon of rap artists is tangential to the actual problem, one of poetic generation. Humans are very good at generating poetry that is consistent with poetic paradigms. For a form such as the sonnet (with syllabic and rhyme constraints), humans will be able to generate satisfactory output with almost 100% accuracy.

Algorithm

Improving upon the baseline, our algorithm will proceed by using depth first search to satisfy the constraints of the poetry object, as follows:

```
while (poetry is unfinished):
      if (poetry is finished) break
      pick some seed line probabilistically based upon n-gram
      writer  with m syllables
      Use random writer to generate x lines with a given syllable
      number m, where each line is semantically consistent with
      the last n words of the seed line
            See if any lines in X meet the rhyming constraints of
            seed line
```

```
                If yes,
                      pair the lines as rhyming pairs
                      update the seed (if appropriate)
      now, combine pairs to generate the output
```

We see this to be an implementation of depth first search. The algorithm will search for the first output that satisfies the constraints of the poetry object. The metric for evaluation is whether the output makes semantic sense, as evaluated by a human. Since the output is intended for human usage, this is a sensible metric.

Example

Thus, consider the following example, define the poetic model to be an Italian Sonnet, which is comprised of an *octave* and *sestet*. Each line is written in iambic pentameter (10 syllables), the octave has rhyme scheme A B B A A B B A, and the sestet has two or three rhyming sounds, arranged in a variety of ways, for simplicity of the example, assume that the sestet has rhyme scheme C D C D C D. The algorithm generates a seed line probabilistically, comprised of 10 syllables (A rhyme). The algorithm then generates the next line probabilistically, in accordance with the constraints (B rhyme). Then, the algorithm must find rhyming pair lines for B and A in succession. It generates some candidate string A', with the necessary number of syllables. If the last words of A and A' do not rhyme, it will back up one level in order to attempt to branch from the penultimate word and find a word that rhymes with A. This paradigm continues until either the poetry object is fulfilled or the work is declared impossible, i.e. no rhyming pairs in accordance with the constraints could be generated.

Next Steps

Moving forward, our priorities have become **a)** finding a way to determine if two words rhyme, and **b)** finding a way to determine the number of syllables in the word. These are two practical issues that we must face in our implementation, both of which will likely entail either importing a dictionary into our model or bootstrapping our system onto an online rhyming engine. After these obstacles are overcome, we will work on tweaking word features in order to improve syntactic consistency, potentially integrating NLP techniques into line evaluation.