Rhymr: Machine-Generated Tumblr Poetry

Sloka Krishnan

Digital poetry—intentional and unintentional—permeates our world. We follow Twitter bots (or accounts presumed to be bots), fascinated by the way in which their disjointed but familiar language speaks to us; we find truth in machine-generated regurgitations of ourselves; we are charmed when our words are shuffled into poetic forms, and stymied when we cannot tell whether a poem was written by a human or by an algorithm. The popularity and prevalence of digital poetry is such that even mainstream organizations like the New York Times have joined in on the trend, turning journalistic excerpts into "serendipitous" haikus.

Interested in the phenomenon of computer-generated poetry and inspired by the recent appearance of Poetweet, a web application that creates formal poetry from a user's tweets, I decided to write a Python script that outputs a sonnet generated from a user's Tumblr posts. The result, rhymr.py, takes a Tumblr username as its only argument. It then:

1) Pulls the user's recent original text posts;
2) Splits these posts into lines of approximately ten syllables each;
3) Finds and pairs rhyming lines; and
4) Prints seven of these pairs in the form of an Elizabethan sonnet (fourteen lines, with the rhyme scheme abab cdcd efef gg).

On Meter

The Elizabethan sonnet has some formal constraints to which the poems generated by this script do not strictly adhere. In its ideal form, a sonnet consists of fourteen lines of iambic pentameter, which means each line would contain five iambs (disyllabic metrical feet that consist of an unstressed syllable followed by a stressed syllable). This script does not search specifically for iambic pentameter, as this would limit results too severely; rather, it ensures that the syllable count of each line totals to roughly ten.

In order to count syllables and find rhymes, the script uses the Carnegie Mellon University Pronouncing Dictionary, accessed through the Natural Language Toolkit for Python. The CMU dictionary, which contains over 133,000 words, splits these words into phonemes and identifies syllabication and lexical stress in a way that is simple, consistent, and machine-readable. One constraint of using this tool, however, is that it is impossible to accurately assess the syllable count of words not found in the dictionary. For this project in its current state, I chose to estimate that all non-dictionary words contain two syllables each; this seems, to me, a solution that better preserves the spirit of Tumblr and poetry alike than the alternative solution of removing all non-dictionary words from the source text. A future amendment to this project could perhaps come up with a smarter estimate of non-dictionary words' syllable count based on word length and/or other factors.

On Rhyme

This script uses the following strategy to determine whether two words rhyme:

1) Find the last stressed syllable in each of the words, and
2) Satisfy the conditions that:
    a) The nucleus (i.e., vowel sound) of this syllable and all subsequent phonemes in the two words are identical, and
    b) The onset of this syllable (i.e., the set of phonemes preceding the nucleus) differs between the two words.

These steps are achieved by parsing each word's entry in the CMU dictionary. Entries are given as strings in which individual phonemes are separated by spaces and nucleus phonemes are each followed by a digit indicating whether the syllable carries no stress (0), primary stress (1), or secondary stress (2). For example, the following code:

```
> transcr = cmudict.dict()
> print transcr['python'][0]
```

would find the pronunciation of the word "python," and thus yield the string:

```
P AY1 TH AA0 N.
```

A "stressed syllable" for the purposes of this project's rhyming algorithm was defined as one indicated by the CMU dictionary to carry primary stress specifically.

On Tumblr

Tumblr is a (micro)blogging platform and social networking site. This script uses the Tumblr API to pull an individual user's text posts. Data is returned in JSON format, details of which can be found in Tumblr's API documentation. Parameters relevant to this project were nested thus:

```
{
      "response": {
            "posts": [
                  {
                        "type": "text",
                        "reblog": "reblog info",
                        "body": "<p>Lorem ipsum dolor sit amet, <a
                        href=”url”>consectetur adipiscing elit</a>.</p>"
                  },
                  {
                        "type": "text",
                        "reblog": "reblog info",
                        "body": "<p>Nam vestibulum consequat justo semper
                        <strong>porttitor</strong>.</p>"
                  },
            ]
```

```
        }
}
```

Currently, the script queries the user's 100 most recent text posts (as opposed to photo, video, audio, etc.). From these, it pulls content only from those posts that were originally authored by the Tumblr user in question (i.e., those that have not been reblogged from somebody else.) Because these posts contain HTML, the script uses BeautifulSoup to quickly and easily retrieve only the text of the posts, stripping them of HTML tags. During this process, the text is also altered in the following ways:

1. All block quotes (which are, after all, unlikely to contain the user's own writing) are removed;
2. Smart quotes are replaced with straight quotes, for the sake of compatibility with the CMU dictionary; and
3. URLs are removed.

On Data, Process, and Next Steps

Early versions of this project used the RhymeBrain API, rather than the above algorithm, to find rhyming words. Calling the RhymeBrain API with a particular word returns JSON data containing all rhyming words contained in the RhymeBrain dictionary. This strategy, while simpler than the algorithm discussed above, proved ineffective due to the high volume of calls the script needed to make to the API in testing the end word of each line against the end of every other line. Calling the RhymeBrain API this many times made execution very slow, and running the program a mere three or four times would very quickly exceed the API's rate limit of 350 requests per hour.

Aside from this change in strategy, however, the script has largely stayed true to its original conception. As mentioned above, a more complex method of syllable counting may be ideal for future use. Other modifications could expand this project's scope to include the generation of additional verse forms, with different line lengths and/or rhyme schemes, as well as the creation of a web application, so that this can run in a public location rather than only from the command line.

Examples

An example sonnet generated from the University of Michigan Student Affairs Tumblr:

Wayne State University Tuesday, November
Awards are currently accepting nominations.
Entering first year students: Remember
Director of Community Relations

of Philosophy, the Spectrum Center, the Lesbian-Gay-Queer
Unions buildings. For more information
costume will Dean Munson sport this year?
This will be followed by a SAPAC presentation

7:30pm Watch the game on the big screen,
at the Sept. 16 meeting, "When
also be available. And, wear your Halloween
2011 Saline Middle School 7190 N

to the first 250 fans through the door!
Services for an appointment or more

And one from the New York Public Library Tumblr:

at the New York Public Library provide
Hurricane Sandy update: NYPL will
the chance to explore education outside
USA – is not in stores until

history of the island and the immigration
Saloon-keepers and street preachers, gypsies and steel-walking
Esteves, and more. More information
with fellow philosopher Jim Holt talking

Hood gets a whole new twist in this retelling.
with "dictionary" in the United States,
On this day in 1758, English-language spelling
a Thursday reopening, but stay tuned for updates.

our Mid-Manhattan Library for a free
is a really important factor. We