Pyambic Pentameter

Generating rhyming and metered poems with Markov chains and NLTK

Kathryn Lingel

How to generate a poem with Python

- What kind of poem?
  - Rhyming lines that have meter
- Generate random text
  - Based on a source corpus
- Make that text sound like a poem
  - NLTK pronouncing dictionary

Poetry 101

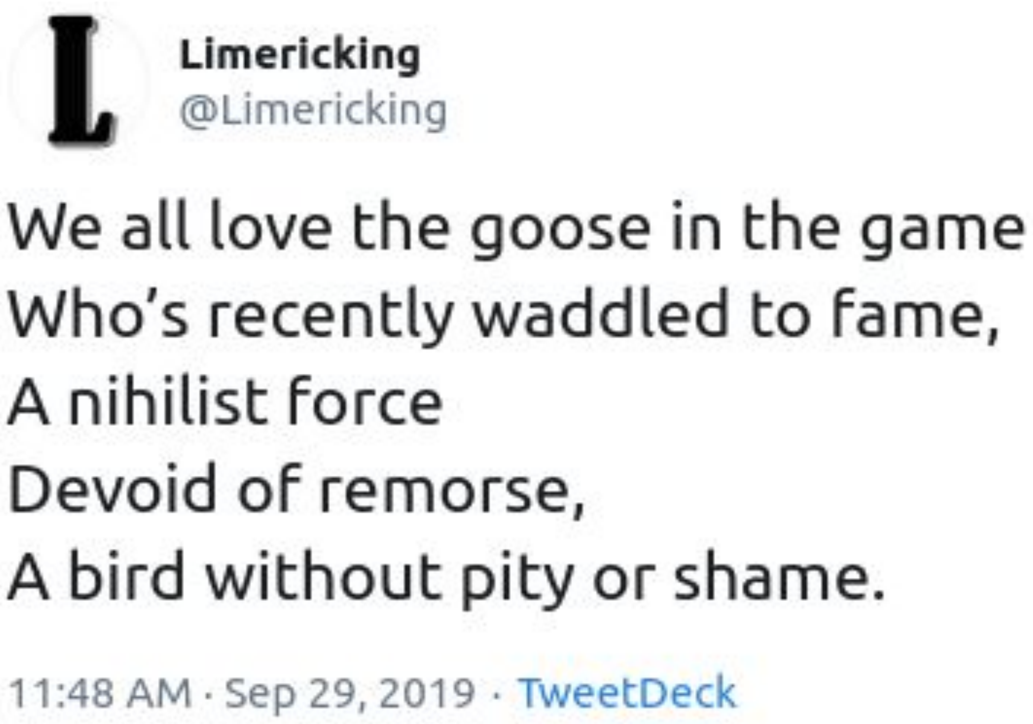
(Disclaimer: I am not a poet)

THeRe WAS A CHIL
D HAD AHOOD HeLIKeDI
T SOMUCh IT WAS
GOOD

KATIE'S 1st POEM FEB. '95

What are we looking for a poem?

- It rhymes!
  - Multiple lines that rhyme with each other
- Meter
  - Consistent stressed/unstressed syllable pattern
- Common formats
  - Sonnet
  - Haiku
  - Limerick



Limericking
@Limericking

We all love the goose in the game
Who's recently waddled to fame,
A nihilist force
Devoid of remorse,
A bird without pity or shame.

11:48 AM · Sep 29, 2019 · TweetDeck

honk

Rhyme

- When multiple words end in the same sound or sounds
- In poetry, usually found at the end of separate lines
- One or more syllables must end the same, depending on emphasis

- hood and good
- bend and intend
- believe and achieve
  - but not dancing and fighting
- remember and December
  - but not October

Meter

- The rhythm of a line
- Described in multiples of “feet”
  - iamb: unstressed-STRESSED
  - trochee: STRESSED-unstressed
  - dactyl: STRESSED-unstressed-unstressed
  - anapest: unstressed-unstressed-STRESSED
  - (and more!)

- “If music be the food of love, play on”
  - iambic pentameter
- “Once upon a midnight dreary while I pondered weak and weary”
  - trochaic octameter
- “‘Twas the night before Christmas and all through the house”
  - anapestic tetrameter

Generating Text

Markov-style 😎

Markov Chain generation

- Word order matters (in English)
- Generate one word at a time
- Pick a word - then pick a word you know is valid after that word
  - Use a corpus to determine what’s “valid”

Markov generation algorithm in Python

- Create a model of the source text
  - For every word that is in the source text, record which words come after it
- Use model + randomness to generate new text
  - Randomly pick a starting “seed” word
  - Given the probabilities from the model, use weighted random to pick the next word
  - If you pick the last word in the text, end early or choose a new seed
  - Repeat until desired length is reached

I am the egg man

They are the egg men

I am the walrus

Goo goo g’joob

—John Lennon

```python
def build_model(source_text):
    list_of_words = source_text.split()
    model = {}

    for i, word in enumerate(list_of_words[:-1]):
        if not word in model:
            model[word] = []
        next_word = list_of_words[i+1]
        model[word].append(next_word)

    return model
```

I am the egg man

They are the egg men

I am the walrus

Goo goo g’joob

i am the egg man they are the egg men i am the walrus goo goo g’joob

```
build_model("i am the egg man they are the egg
men i am the walrus goo goo g'joob")
```

```
{'i': ['am', 'am'],
 'am': ['the', 'the'],
 'the': ['egg', 'egg', 'walrus'],
 'egg': ['man', 'men'],
 'man': ['they'],
 'they': ['are'],
 'are': ['the'],
 'men': ['i'],
 'walrus': ['goo'],
 'goo': ['goo', "g'joob"]}
```

```python
import random

def markov_generate(source_text, num_words=20):
    model = build_model(source_text)
    seed = random.choice(list(model.keys()))
    output = [seed]
    for i in range(num_words):
        last_word = output[-1]
        next_word = random.choice(model[last_word])
        output.append(next_word)
        if next_word not in model:
            break
    return ' '.join(output)
```

```
markov_generate("i am the egg man they are the egg men i am the walrus goo goo g'joob")
```

```
"the egg men i am the egg man they are the walrus goo goo goo g'joob"

"men i am the walrus goo g'joob"

"men i am the egg men i am the egg men i am the egg man they are the egg man"

"goo goo goo goo goo goo goo g'joob"
```

An Incomplete List of Potential Enhancements

- More training text
- Better lookahead when picking a word
  - “Markov order”
- Be smarter about which words we pick

NLTK

Natural Language Toolkit

Features of NLTK

- Sentence diagramming
- Word tokenization
- Sentiment analysis
- Sample corpuses
  - Names
  - Cities
  - Books
  - Tweets
- Pronunciation

Carnegie Mellon Pronouncing Dictionary

or `nltk.corpus.cmudict`

- Hello
  - `[['HH', 'AH0', 'L', 'OW1'], ['HH', 'EH0', 'L', 'OW1']]`
  - “hal-LO”, “hel-LO”
- Python
  - `[['P', 'AY1', 'TH', 'AA0', 'N']]`
  - “PY-thon”
- Archipelago
  - `[['AA2', 'R', 'K', 'AH0', 'P', 'EH1', 'L', 'AH0', 'G', 'OW2']]`
- Kathryn, Katharine(, Catherine, Katherine)
  - `[['K', 'AE1', 'TH', 'R', 'IH0', 'N']]`

Generating Poetry

Putting it all together

Making it Rhyme

- Create a “rhyme model”
  - When going through the text, look up every word in the pronouncing dictionary
  - Slice off the pronunciation starting at the last emphasized vowel
  - Organize all words by rhyme
- Pick a rhyme seed
  - And from there, two (or more) Markov seeds that rhyme
- Generate lines backwards

Rhymes in “I am the Walrus”

```
{('OW1',): ['grow', 'row', 'poe', 'ho'],
  ('AH1', 'N'): ['run', 'gun', 'sun'],
  ('AH1', 'M'): ['from', 'come'],
  ('AY1',): ['i', 'eye', 'sky', 'fly', 'sty'],
  ('UW1',): ['you', 'to', 'goo', 'do'],
  ('EH1', 'T'): ['get', 'let'],
  ('IY1',): ['he', 'hee', 'see', 'me', 'we'],
  ('AO1', 'R'): ['for', 'your'],
  ('AE1', 'N'): ['man', 'an', 'van', 'tan'],
  ('IH1', 'T', 'IY0'): ['city', 'pretty']})
```

Giving it Meter

- “Guess and check” or “backtrack”
- Describe the meter of the line: `"0101010101"`
- Transform a word into its “meter fingerprint”
  - “python” becomes `"10"`
  - “hello” becomes `"01"`
  - “archipelago” becomes `"10101"`
- When choosing your next word, reject a word if it doesn’t make your output fit the meter
  - `"0101010101"` + “python” = ❌
  - `"0101010101"` + “archipelago” = generate a pattern to fit `"01010"` + “archipelago”
- If no options fit the meter, delete the last word you chose and try again with a different option
  - `"0101010101"` + “hello” = generate a pattern to fit `"01010101"` + “hello”

Extra tips and tricks

- Words that don’t appear in the CMU dictionary: guess the syllables!
    - Vowel groups
    - `Num2word`
    - Punctuation
- Poem DSL

Live Demo!

What could go wrong?

Acknowledgements

- Inspired by @pentametron 🤖
- Speaker mentor: Your Name Here???
- Craigslist data set: Britt Gresham @demophoon
- First-time speaker resources
- Pyladies and PuPPy

Thank you!

I am @hartknyx

Neverending poetry at https://poems.katlings.net/

Source and slides at https://github.com/katlings/pyambic-pentameter

Recommended Resources

- A discussion of poetry in different languages http://stories.schwa-fire.com/language_of_poetry
- The intro-to-CS assignment that taught me about Markov generation https://www.cs.hmc.edu/twiki/bin/view/CSforAll/MarkovText1
- Pycon Israel 2016: “Poetry in Python: Using Markov Chains to Generate Texts” by Omer Nevo https://www.youtube.com/watch?v=yyNTjDkQQEk
- The Mechanical Muse, a New Yorker article about machine-generated sonnets https://www.newyorker.com/culture/annals-of-inquiry/the-mechanical-muse

just to be sleeping everybody seems
and know i'm telling you became a sign
and dig a smoke and understand his dreams
it down upon her words to last it's mine

whenever you can never heard the smiles
proceeded to pretend that weight a yes
the sight of yellow matter much in miles
it does it feels good news today the press

this is a bad the west behind and shot
if we will never win and died in years
the one is high or low that way it's not
it ain't the walrus gumboot he appears

is to the end the dirt and finger he
it can be an unpleasant scene and bee