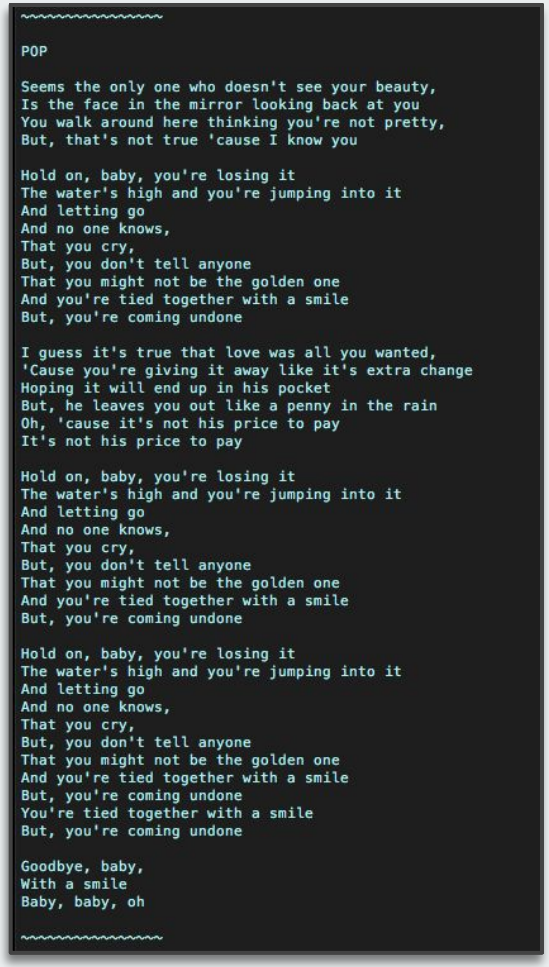
Song Lyric Corpus

Alan, Dante, Jacob, Nick, and Shirley



```
~~~~~~~~~~~~~~~~~~

POP

Seems the only one who doesn't see your beauty,
Is the face in the mirror looking back at you
You walk around here thinking you're not pretty,
But, that's not true 'cause I know you

Hold on, baby, you're losing it
The water's high and you're jumping into it
And letting go
And no one knows,
That you cry,
But, you don't tell anyone
That you might not be the golden one
And you're tied together with a smile
But, you're coming undone

I guess it's true that love was all you wanted,
'Cause you're giving it away like it's extra change
Hoping it will end up in his pocket
But, he leaves you out like a penny in the rain
Oh, 'cause it's not his price to pay
It's not his price to pay

Hold on, baby, you're losing it
The water's high and you're jumping into it
And letting go
And no one knows,
That you cry,
But, you don't tell anyone
That you might not be the golden one
And you're tied together with a smile
But, you're coming undone

Hold on, baby, you're losing it
The water's high and you're jumping into it
And letting go
And no one knows,
That you cry,
But, you don't tell anyone
That you might not be the golden one
And you're tied together with a smile
But, you're coming undone
You're tied together with a smile
But, you're coming undone

Goodbye, baby,
With a smile
Baby, baby, oh

~~~~~~~~~~~~~~~~~~
```

What is this corpus?

- Robust list of 3926 song lyrics across four genres
  - Country, rock, pop, rap
- Can retrieve songs of a specific genre
- Can get the stanzas, lines and words of a song

Potential Usages

- Generate songs using Markov chains that follow rhyme patterns
- Statistics on certain genres
  - Most common words
  - Average song/stanza/line lengths
  - Most frequent rhymes
- Training classifier to recognize genre of lyrics
  - Rhyme pattern is just one of many features to extract from this dataset
  - Output space is {Rock, Rap, Country, Pop}

Step 1.1: Gather Artist Set

Using **SpotiPy**, we were able to retrieve Spotify's top *N* songs from their top *M* artists per specified category

Categories: **Rock**, **Rap**, **Country**, and **Pop**

We gather these (artist,song) pairs and pass them into **PyLyrics** to receive our songs

Step 1.2: Gather songs

We used **PyLyrics**, and API where we can retrieve lyrics hosted on *lyrics.wikia.com* via Python

PyLyrics will return lyrics based on the following input:

- Name of Artist, and
- Name of Song

Step 2: Compile dataset as text files

Upon each successful call to **PyLyrics**, we write the lyrics to a .txt file

lyrics.wikia has a convention where there is a newline (‘\n’) character between every line of lyrics, and two newlines between every stanza

ex. “Lorem ipsum**\n**Dolor sit amet,**\n\n**in velit iudico**\n**vivendum sea”

To mark the boundary between songs, we add a special token after every song when writing to the text file: “**\n\n**<SONG_BOUNDARY>**\n\n**”

Rhyme Pattern

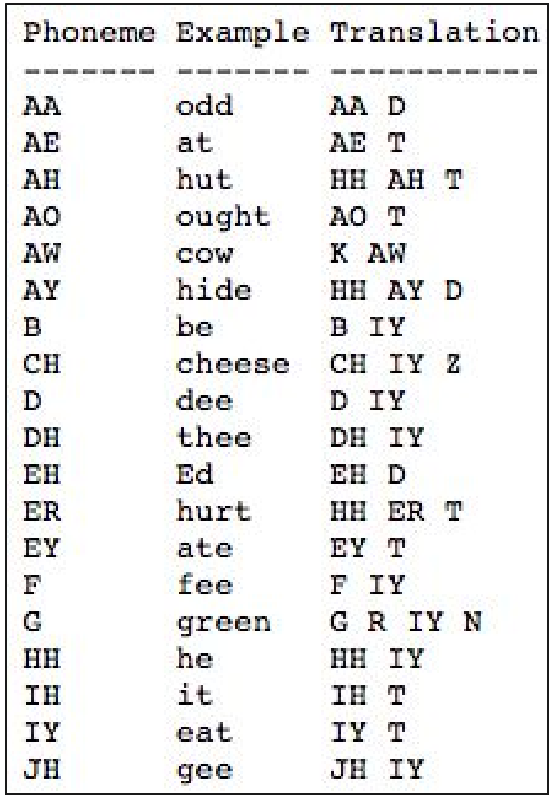
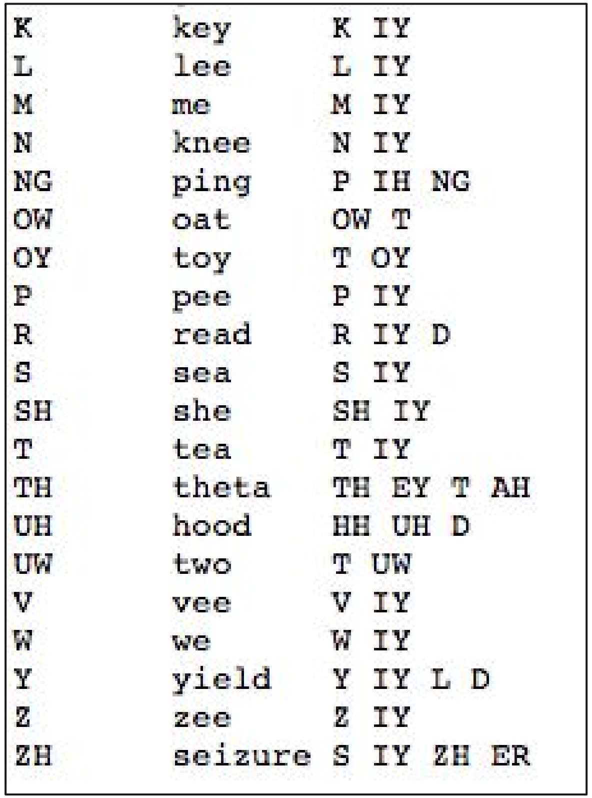
An idea for something to do with this dataset: get rhyme patterns for each stanza!

A rhyming pair is two words that have the same last syllable

How to determine last syllable? Use **CMUDict**!

Idea: Get last token from each line, feed into **CMUDict**, get phonetic representation, get last syllable by checking for */d* regex in the phoneme string, and check for equality between the lists of phonemes

Phoneme Representation

Phoneme	Example	Translation
AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	Ed	EH D
ER	hurt	HH ER T
EY	ate	EY T
F	fee	F IY
G	green	G R IY N
HH	he	HH IY
IH	it	IH T
IY	eat	IY T
JH	gee	JH IY
K	key	K IY
L	lee	L IY
M	me	M IY
N	knee	N IY
NG	ping	P IH NG
OW	oat	OW T
OY	toy	T OY
P	pee	P IY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	theta	TH EY T AH
UH	hood	HH UH D
UW	two	T UW
V	vee	V IY
W	we	W IY
Y	yield	Y IY L D
Z	zee	Z IY
ZH	seizure	S IY ZH ER

Rhyme Pattern Algorithm

```
Input = stanza, T = list of line-terminal tokens in stanza, R: (String, String) -> Bool
cur ← 0, C = [ ]
for i in 0..<length(T)
    If C[i] has no label → C[i] = cur
    for j in i+1..<length(T)
        if C[j] has no label and R(T[i], T[j]) → C[j] = cur
    cur ++
Output = C
```

NLTK Wrapper

An NLTK-based CorpusReader for the corpus

Easily access songs → stanzas → lines → words from the text files

Uses StreamBackedCorpusView to prevent loading entire corpus in memory

Helper Song and Stanza objects

Improvement: Add Rhyme annotations (either on file or through a simplified API call)

NLTK Wrapper: Objects

SongCorpusReader

Attributes: tokenizer WordPunctTokenizer(), fileids, root, encoding UTF-8

Functions: raw(), words(), songs(), _read_song_block(), _read_stanza_block()

Song

Attributes: title String, list of Stanza objects

Functions: lines(), words(), add_stanzas()

Stanza

Attributes: list of String objects representing lines

Functions: N/A

NLTK Wrapper: Attributes and Functions

reader = SongCorpusReader()

● reader.songs()	list of 3907 Song objects
● reader.words()	list of 1495039 words
● reader.songs(‘rock.txt’)	list of 1142 rock Song objects

song = reader.songs()[0]	Tim McGraw’s “Welcome to the Club”

● song.lines()	list of 26 strings representing a line
● song.words()	list of 187 words
● song.stanzas	list of 4 Stanza objects

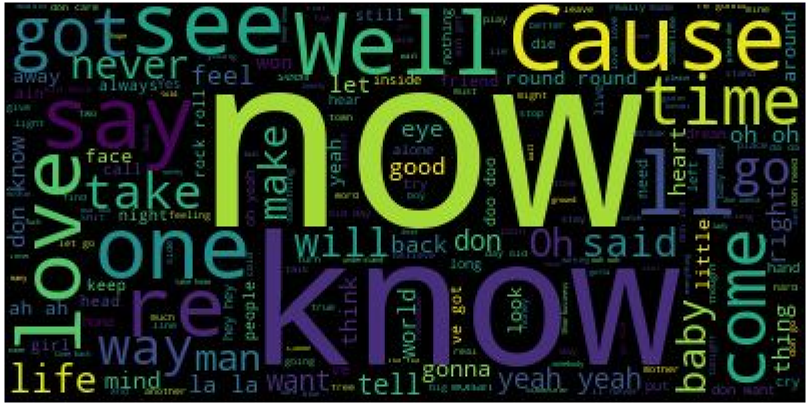
NLTK Wrapper: Algorithm

```
_read_song_block():
    While not at the end of stream:
        Make a Song object
        While not at <SONG_BOUNDARY> or end of stream:
            Add result of _read_stanza_block() to Song
        Add Song to a list
    Return list of Song objects
_read_stanza_block():
    While not at an empty line (stanza delimiter) or end of stream:
        Add current line to a list of lines
    Return Stanza(lines)
```

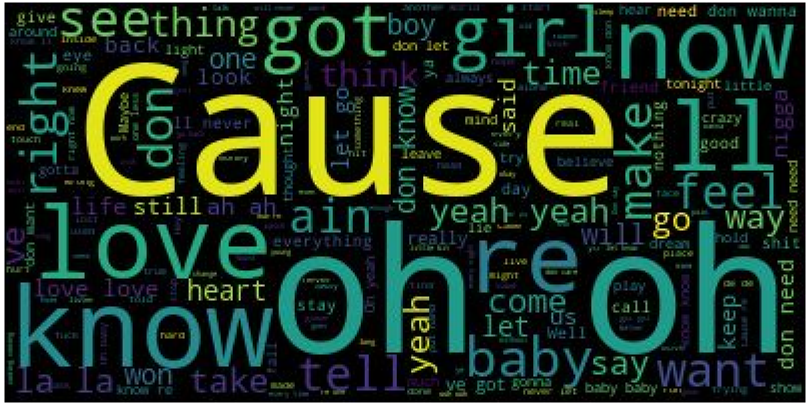
Stats

Feature	Rock	Rap	Pop	Country
# Songs	1142	829	932	1004
# Stanzas	8174	5904	8182	6941
# Lines	38216	50949	45208	35786
Size (MB)	1.22	2.04	1.55	1.30

Word Clouds



ROCK



POP

Possible Improvements

- Including song names
    - While we were classifying and parsing each of the songs originally, we did not include the names of the songs.
- Attach artist information to songs
    - This would allow us to compare songs among different artists.
- Include song length (in terms of time)
    - Could be useful for analyzing song density in terms of lyrics or line count over time
- Rhythm annotations available from the original source, when applicable
    - Some songs might have rhythm annotations available that might be useful for rhythm analysis.
- Encoding errors (we used UTF-8)

Thank you!

Got questions or comments? Ask us!