Joy Division
New Order

Analyzing Band Lyrics for
Classification and Prediction

Album Selection


Unknown Pleasures
1979


Closer
1980

Movement
1981


Power, Corruption and Lies
1983

Low-Life
1985


Brotherhood
1986


Technique
1989


Republic
1993


Get Ready
2001

Waiting for the Sirens' Call
2005


Singles
2005


Lost Sirens
2013

Credit: New Order Discography, Genius, and Wikipedia

Data Gathering

- Official Genius API
  - Create an account to get started
  - Verify email
  - Get authorization token (authtoken)
  - Include authtoken with each API request

What is a REST API?

Start Small

- Search Request with authtoken in HTTP Request header

  `http://api.genius.com/search?q=song-title-term`

  Returns a response JSON object with 10 hits.

Genius API Documentation

Borrow a Ladder

- LyricsGenius: A Python client for the Genius.com API
  - Install the client and it simplified the task to (1) `declare track` (2) `declare artist` per call to the `genius.search_song` function
  - Including the authtoken is transparent to the call
  - The response is serialized to disk as a JSON file

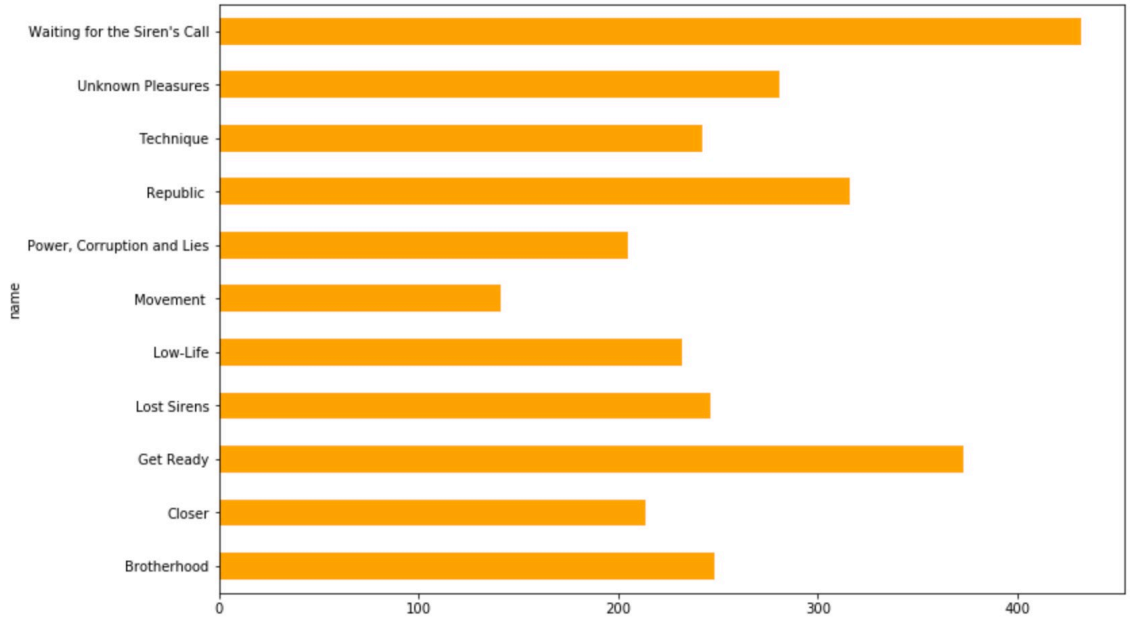
JSON => SQLite

- The per-track JSON documents begin with a header record that conceptually is the main table for a small SQL database
- A single child table will contain the track information and song lyrics
- The song lyrics will preserve the formatting provided by Genius
- Pandas `read_sql_query` enables data access in Jupyter NB

Data Prep

- Elected to split each song lyric on newline to have a larger DataFrame
  - Complete at 2,930 rows
- Generated song lyrics metrics: length, length (unique term)
- Added lemmatization features from the `SpaCY` NLP library
- Discovered that track “5-8-6” returned a legal document 🤷‍♂️

Data In-place



Model: LinearRegression/CV

- Fitting a linear regression on lemmatized and count vectorized song lyrics `(x)` and song title length `(y)` would produce a (very low and best case) positive score when the feature count was reduced more than 60%

`CountVectorizer(max_features)` < 500

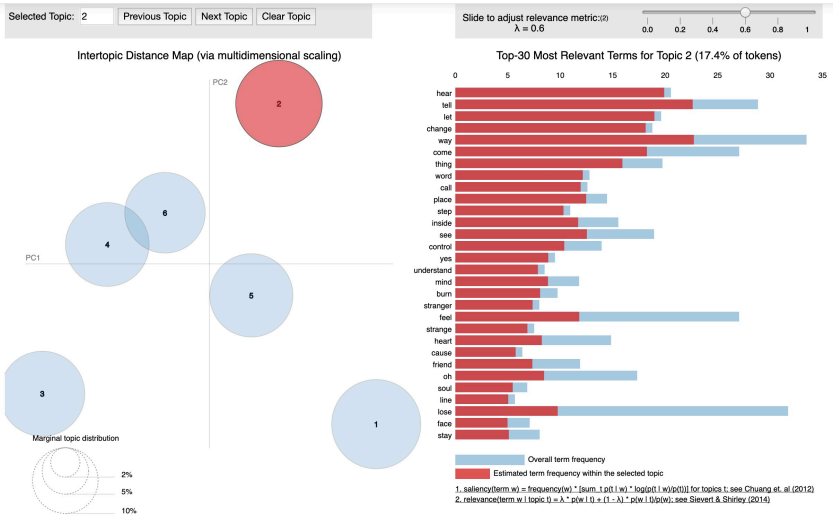
know 144	get 135	like 134	time 112	want 89
go 83	day 80	right 80	need 79	way 78

Model: LinearRegression/TfidF

- Fitting a linear regression on TfidF vectorized song lyrics `(x)` and unique-mapped album counter `(y)` would produce a more acceptable score of 0.7100
  - `SKLEARN.TfidfVectorizer` was responsible for removing STOP_WORDS
  - Parameter `ngram_range` was default at `(1,1)`

Model: TfidF/LDA

- `LatentDirichletAllocation` with `n_components` 6 gave good separation when viewed by `pyLDAvis`



Continued: LDA, 6 Topics, Top 10 terms

Topic

day look go take leave world hand get to away

Topic

time believe people wanna long love way trust road like

Topic

like try think life fall man far stand feel remember

Topic

want lose will because kind free wake hang home real

Topic

know right need good end wrong care live get time

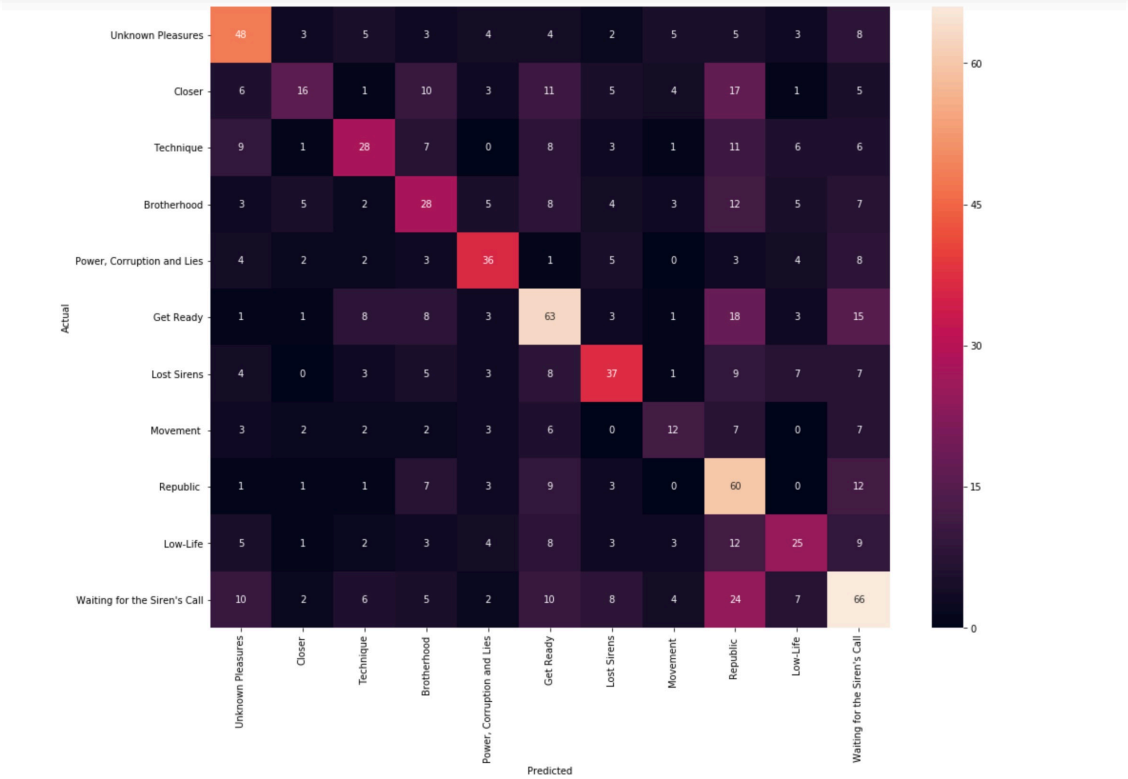
Topic

way tell hear let come change thing see place word

Model: Miscellany

- MultinomialNB
- LinearSVC ⭐
- LogisticRegression
- RandomForestClassifier

Model: Score 'Predicted Lyric' on SVC



Next Steps

- Small: ???
- Medium: Generative LSTM
  - Single-serving site that generates a "card" of possible song titles for your selected album/track
- Are-You-Out-of-Mind: Prism