Lyric-based Playlist Generation

Creston Brooks, Henrique Schechter Vera, Elaine Wright

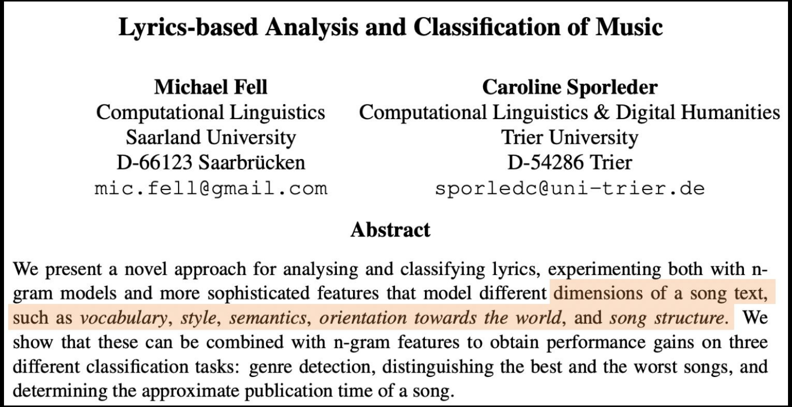
May 7, 2021

Background and Motivation

- **Industry song recommendations (e.g., Spotify) based on user data (“collaborative filtering”), not necessarily lyrical content**
- **“Collaborative filtering” potentially not optimal**
    - Popular songs get recommended, making them more popular
    - Less popular songs might not get recommended even if they would match a user’s taste very closely
- **How much does lyrical content matter in effective playlist generation?**
- **Text is easier to process than audio: more efficient, less abstract**

Related Work

- **“Lyrics-based Analysis and Classification of Music”, Fell and Sporleder (2014)**
    - Motivations: lyrics easier to process than audio data; lyrics a “proxy for the melodic, structural and rhythmic properties of the audio...melody and rhythm...[can] be traced in the stress pattern of the text; brain processes text and audio independently–both contribute to the appreciation of a song
    - Use: finding “best”/“worst” songs, genres, release dates



Lyrics-based Analysis and Classification of Music

Michael Fell
Computational Linguistics
Saarland University
D-66123 Saarbrücken
mic.fell@gmail.com

Caroline Sporleder
Computational Linguistics & Digital Humanities
Trier University
D-54286 Trier
sporledc@uni-trier.de

Abstract

We present a novel approach for analysing and classifying lyrics, experimenting both with n-gram models and more sophisticated features that model different dimensions of a song text, such as *vocabulary*, *style*, *semantics*, *orientation towards the world*, and *song structure*. We show that these can be combined with n-gram features to obtain performance gains on three different classification tasks: genre detection, distinguishing the best and the worst songs, and determining the approximate publication time of a song.

Related Work

- **“DeepPlaylist: Using Recurrent Neural Networks to Predict Song Similarity” (Balakrishnan and Dixit, 2016)**
  - General NN architecture (explained later)
- **“Combining Content and Sentiment Analysis on Lyrics for a Lightweight Emotion-Aware Chinese Song Recommendation System” (Chen and Tang, 2018)**
  - Use tf-idf (importance of word) and lexicon-based sentiment analysis
- **“Lyrics or Audio for Music Recommendation?” (Vystrčilová and Peška, 2020)**
  - Limit scope to recommend/evaluate using one “seed” song

Dataset and Tools

- **Spotify Million Playlist Dataset**
  - 1 mil. playlists, 2 mil. tracks
  - 300,000 artists
- **Genius API / lyricsgenius**
  - Song lyrics with annotations
- **Featurization**
  - Python, shelve, re, gensim, nltk, sentiwordnet, num2words, syllables
- **Neural Network**
  - Pytorch, pandas, sklearn, numpy
- **Evaluation**
  - Scipy, numpy, random, langdetect

Approach

- **Represent songs as vectors of features**
- **Score features, recommend songs with similar scores**
  - Vocabulary
  - Structure
  - Orientation
  - Style
  - Semantics
- **Song preprocessing**
  - URI, dictionary
  - Songs in English
  - Parsing, stopwords, tokenization, remove labels/punctuation (varies by feature)
- **Recommendation**
  - Neural network: be able to get probability songs are in a playlist together, recommend songs most likely to be in a playlist with all others (assume independence of songs)
  - Cosine similarity: recommend songs most similar to seed song(s): either most similar to mean of seed songs, or most similar to any seed song
- **Challenges**
  - Size of data: scalability, processing time

List of Features

Vocabulary

- Taboo, offensive tokens
- Average word frequency
- Syllables/word
- Proportion of proper or common nouns
- Proportion of plain, comparative, superlative adjectives
- Proportion of noun, verb, adjective tokens

Structure

- Proportion of lines that have an exact/fuzzy match for structure alignment
- Structure of song parts
- Title contained, not contained

Orientation

- Person (1st/2nd/3rd)
- Distribution of verb tenses

Style

- Lines/song
- Words/song
- Syllables/second
- Proportion of consecutive (1 or 2 apart) rhyming lines

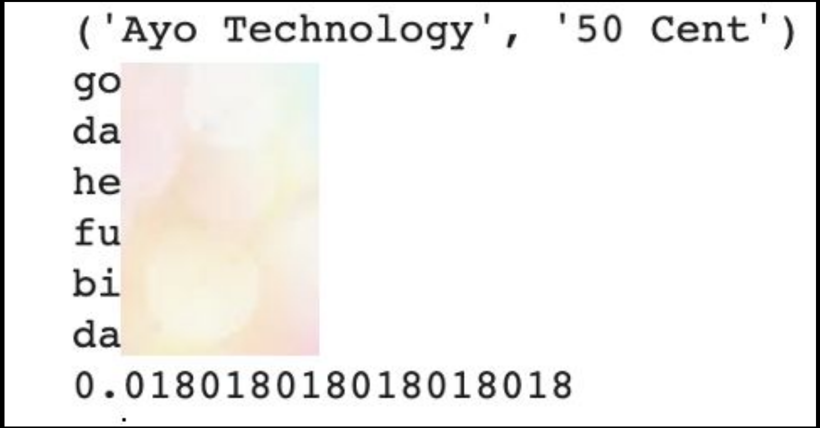
Semantics

- Doc2vec
- Sentiment of important words

Vocabulary

- **Taboo, offensive tokens (1029 total)**
- **Average word frequency**
- **Syllables/word**
    - Lexical complexity correlated to syllables
- **Noun and adjective types**
    - Proportion of proper or common nouns
    - Proportion of plain, comparative, superlative adjectives
- **Token POS ratio**
    - Proportion of noun, verb, and adjective tokens

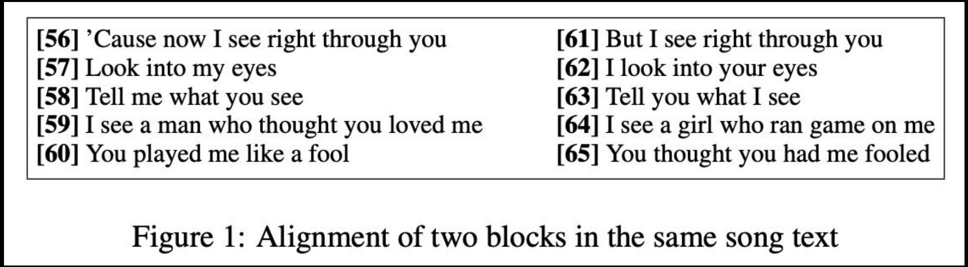
$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$



```
('Ayo Technology', '50 Cent')
go[redacted]
da[redacted]
he[redacted]
fu[redacted]
bi[redacted]
da[redacted]
0.018018018018018018
```

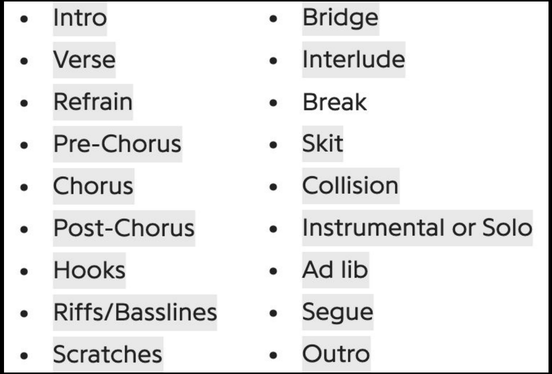
Structure

- **Title contained**
    - Unique when it is not
- **Repetitive structure**
    - Proportion of lines exhibiting an exact/fuzzy match
- **Structure of parts**
    - Edit distance from "standard structure"



[56] 'Cause now I see right through you	**[61]** But I see right through you
[57] Look into my eyes	**[62]** I look into your eyes
[58] Tell me what you see	**[63]** Tell you what I see
[59] I see a man who thought you loved me	**[64]** I see a girl who ran game on me
[60] You played me like a fool	**[65]** You thought you had me fooled

Figure 1: Alignment of two blocks in the same song text



- Intro
- Verse
- Refrain
- Pre-Chorus
- Chorus
- Post-Chorus
- Hooks
- Riffs/Basslines
- Scratches
- Bridge
- Interlude
- Break
- Skit
- Collision
- Instrumental or Solo
- Ad lib
- Segue
- Outro

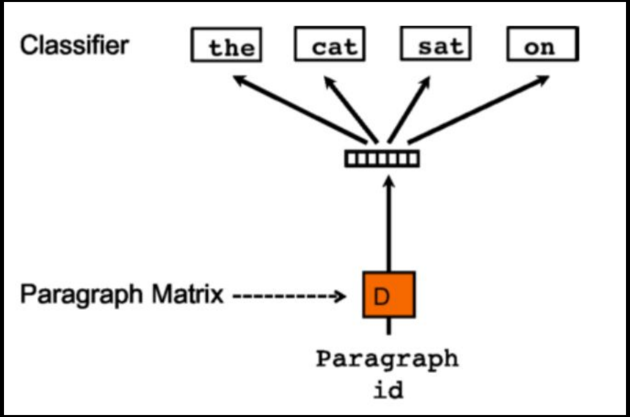
Orientation

- **Pronouns**
  - e.g. “egocentric” monologue vs. flirtation with lover
    - Kanye West - Power vs. Frankie Valli - Can’t Take My Eyes Off You
  - [y’all, thy, youse, whatcha, yeerselves…] and other fun pronouns
- **Verb tense**
  - Proportion of verb tokens that are past, present, and future

Style

- **Lines/song**
- **Words/song**
- **Syllables/second**
    - Proxy for song tempo
- **Rhyming lines**
    - Proportion of lines whose last word is a perfect rhyme with either of the following two lines' last words
    - Attempted to compare IPA vowels of nearby words to detect near-rhyme, but was too slow (~5 sec per song)

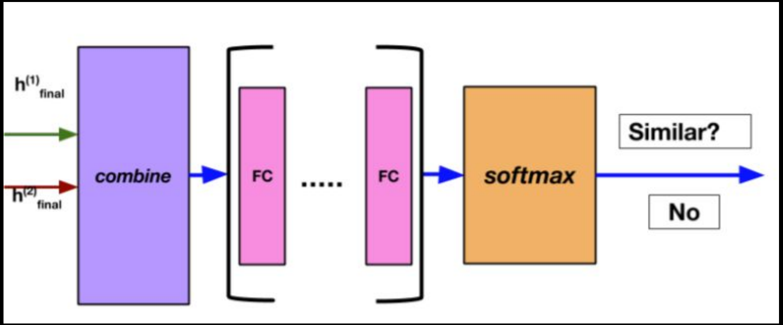
Semantics

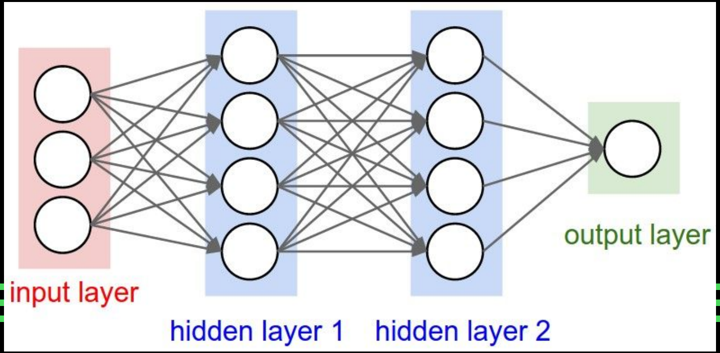


- **Doc2Vec**
  - "Ignore the context words in the input, but force the model to predict words randomly sampled from the paragraph in the output"
  - Representation is vector that best predicts words in paragraph
- **Sentiwordnet tf-idf**
  - tf-idf(term i, doc j) = $tf_{i,j} * [1 + \log((1 + N)/(1 + df_i))]$
  - Sentiment of word: (positive, negative) averaged over meanings
  - Sum over 5 most important words to get document sentiment

Neural Network

- **Given inner product of song features, what is the probability that the two songs appear in a playlist together?**
  - ~32k/350k pairs match (844 unique songs)
- **Architecture:**
  - Inner product of songs
  - Fully-connected
  - ReLU activation
  - Sigmoid output
  - Hyperparameters: layers, hidden neurons, learning rate, epochs
- **Insufficient data?**
  - Only over a dozen playlists' worth
  - ~0.515 accuracy on test data





Cosine Similarity

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- **Computed mean similarity for all pairs songs that co-occur in playlist (~4%, n = ~32,000)**
- **Computed mean similarity for all pairs of songs (~0.01%, n = ~350,000)**
  - Songs that co-occur in playlist have significantly higher cos. similarity
- **For playlist generation given a seed song, compute similarities of seed to every other song, return n highest similarities**

Evaluation

- Iterating over all songs in our database as seeds, we generated playlist of varying lengths (n = 1, 5, 10...) and achieved a success rate of ~24%, meaning that 24% of songs we generated for a seed song actually co-occurred with that seed song in some playlist.
- Randomly generated playlists of varying lengths achieved a success rate of ~9.5%

Evaluation Breakdown

- With further testing, we find that playlist generation is still **~23% successful** even without doc2vec
- Performing playlist generation while isolating each feature, we also estimate which features are most important

Best Features

1. Syl / sec, syl / word
2. Total lines / words
3. Sentiment analysis
4. Overall POS proportions
5. Pronoun proportions

Conclusion and Future Work

- With current database, playlist generation success vs. random chance: **24% vs. 10%**
- For more conclusive evaluations of individual features and more testing, expanded database is necessary
    - Process is streamlined but will require several hours of processing
- More data will also better illuminate individual feature performance, allowing us to optimize components and focus on best-performing features
- Lots of other seemingly promising features were too computationally costly, some in the paper
- Eventual synthesis with audio-based analysis for maximum success

Questions?

🎵

Roles and Responsibilities

Elaine: Structure parsing of songs and preprocessing to standardize names of structure parts (re); calculating a song’s deviation from standard structure as edit distance (Levenshtein); exact and fuzzy alignment of related lexical/lyrical structures within a song (fuzzywuzzy); proportion of rhyming lines (pronouncing, num2words); title contained in song lyrics, exact and fuzzy (fuzzywuzzy); proportions of types of nouns, verbs, and adjectives (nltk); proportion of noun, verb, and adjective tokens (nltk)

Roles and Responsibilities

Henrique: scraping lyrics from song names (lyricsgenius, json); putting all data (names, lyrics, urls, features, etc.) into python objects and saving to disk/drive (shelve); putting together all features and writing the song recommender (scipy, numpy); preparing examples/data for, creating, training, and evaluating the neural net (torch, pandas, sklearn); lyric preprocessing and tokenization for most features using regular expressions, stopword removal, etc. (re, gensim, nltk); sentiment analysis feature using tf-idf (sentiwordnet); training/creating doc2vec model (gensim); 1st, 2nd, 3rd person orientation feature; getting and analyzing results and performance of model w/ and w/o doc2vec, and of different individual features

Roles and Responsibilities

Creston: lyrical pre-processing and tokenization across features, (num2words, regular expression, etc.) Lexical complexity feature spanning complete lexicon. Quick syllable estimator (syllables) to compute syl/sec and syl/word without dict reference. Lexical lengths of lyrics. Adapting and augmenting CMU list of offensive words, resulting in list of 1029 “taboo” words, implementing taboo feature. Conversion to IPA vowels (eng_to_ipa) for near rhyme measure. Compilation of all songs that co-occur in playlists for training and eval. Data compression of song pair uri’s for achievable search/storage. Creation of dictionaries to go between compressed uri’s, lyric indices, and lyrics. English language detection (langdetect) and threshold setting. Final song recommender, testing, and results evaluation.