Semantic Analysis of Popular Song Lyrics From 2010 to 2024.

By Ren Tao

Introduction

Context: Song lyrics reflect the cultural, emotional, and thematic trends/ideas of their time. Since pop culture is so mainstream and arguably listened to more by everyone, the lyrics of pop songs should also be reflective of the ideas of the time period in which they were released.

Hypotheses:

- **H1:** Songs from similar years will have higher lyrical similarity.
- **H2:** Songs will cluster according to their lyrical, semantic meaning,
- **H3:** Pop lyrics have become more homogeneous over time.

Objective: To empirically analyze clustering and similarity patterns across Billboard's Top 10 songs in each year from 2010 to 2024.

Motivation: Looking into how thematically song lyrics cluster together semantically, can potentially reveal aspects of the 2010's popular culture. This is really interesting especially in the context of the space-time compressed age we live in, where certain mainstream ideas are globalized and sensationalized.

Methodology

Dataset:

- Billboard Top 100 songs from 2010 to 2024. Scraped from Wikipedia.
- Lyrics obtained via web scraping on azlyrics.com, and at times collected manually sources through LyricFind and MusixMatch.

Overview:

Phase 1: Java-Based Scraping and Vectorization

- Scraping: Java (Parser, URLGetter)
- Cleaning: Java (regex, string methods)

- Vectorization: Java (TF-IDF via Corpus and VectorSpaceModel)
- Output: CSV (using OpenCSV library)

Phase 2: Python-Based Analysis and Visualization

- Data Analysis: Python (Pandas, Scikit-learn)
- Dimensionality Reduction: Python (TruncatedSVD)
- Cosine Similarity: Cosine similarity matrix computed across all song pairs
- Clustering: Python (KMeans, AgglomerativeClustering, OPTICS)
- Visualization: Python (Matplotlib, Seaborn)

Detailed Methodology:

Phase 1: Java-Based Scraping and Vectorization

- Scraping Song Titles and Artists to find the lyrics of: Titles and artists scraped from Wikipedia pages on Billboard Year End Top 100 songs (2010 – 2024), using classes LyricFetcher and Parser (reused from HW3). Logic mainly consists of using Regex to extract out song titles and artists names via the raw HTML content received from the GET request to https://en.wikipedia.org/wiki/Billboard_Year-End_Hot_100_singles_of_[year]

- Retrieving Lyrics: Retrieval from https://www.azlyrics.com/lyrics/[artist]/[song].html also using LyricFetcher and Parser, matched format of the website and immediately saved scraped lyrics to lyricsRaw.csv to avoid duplicated requests. Added Thread.sleep() between requests to avoid bot detection. Manually retrieved skipped songs where lyrics could not be retrieved via web scraper due to bad [artist]/[song] format inaccuracies.

- Preprocessing & Cleaning Lyrics: Used LyricProcessor LyricCleaner classes to remove punctuation, lowercase all lyrics, apply stopword removal using NTLK's stopwords list (sourced on GitHub). Also removed duplicated consecutive words(typically found in the chorus/riff, this was to ensure the lyrical meaning is not too bogged down by repetition of words). Saved cleaned lyrics in lyrics.csv

- Logistical set up for TF-IDF vector analysis: Used the lyrics.csv file and to create .txt file per song, grouped by year, this is stored in the directory "data/songs/" to be used to generate documents later on. Done so in LyricProcessor class.

- Vectorizing Lyrics: Built a corpus in VSMBuilder class by iterating through all .txt song files and generated a TF-IDF Vector Space Model using the Corpus object, using the Corpus and VectorSpaceModel class provided in HW4. Saved result in vectors.csv containing year, artist, song title information along with its TD-IDF vector.

Phase II: Python-Based Analysis and Visualization

- Loading and Organizing Data: Imported vectors.csv into a pandas DataFrame in Google Colab. Parsed the TF-IDF vector columns, and extracted respective metadata like year, songs, and artist names.
- Dimensionality Reduction: Used TruncatedSVD from sklearn.decomposition to reduce the TF-IDF vectors from high-dimensional space (thousands of terms) to 100D for efficient clustering and similarity analysis. Additionally reduced to 2D for ease of plotting and visualization in a 2D graph also using the SVD.
- Outlier Filtering: Used Euclidean distance in the 2D space to manually identify and filter out strong outlier songs that were very far from the main cluster. These songs were analyzed later separately for their distinctive lyrics.
- Cosine Similarity Analysis: Computed the cosine similarity matrix across all songs using sklearn.metrics.pairwise.cosine_similarity. Used the matrix to analyze average similarity between songs across different years (year-to-year similarity), and additionally to identify the most similar and most different song pairs based on their lyrical content.
- Clustering Techniques: Applied several clustering algorithms to the vectors to detect potential clustering among songs:
    - KMeans (k=3): Fixed-group clustering. Determined k=3 using elbow method. Analyzed cluster sizes, top words per cluster, and most central songs per cluster.
    - Agglomerative Clustering: Visualize hierarchical similarity and find any underlying structure not well captured by KMeans.
    - OPTICS: Density-based clustering to detect organically separated groups and noise without assuming a fixed number of clusters.
- Visualization: Used matplotlib and seaborn to produce:
    - A 2D semantic map of all songs colored by year
    - Clustered scatterplots for KMeans, Agglomerative, and OPTICS
    - A heatmap showing average cosine similarity between songs of different years
    - Bar charts of top words per cluster and cluster size summaries

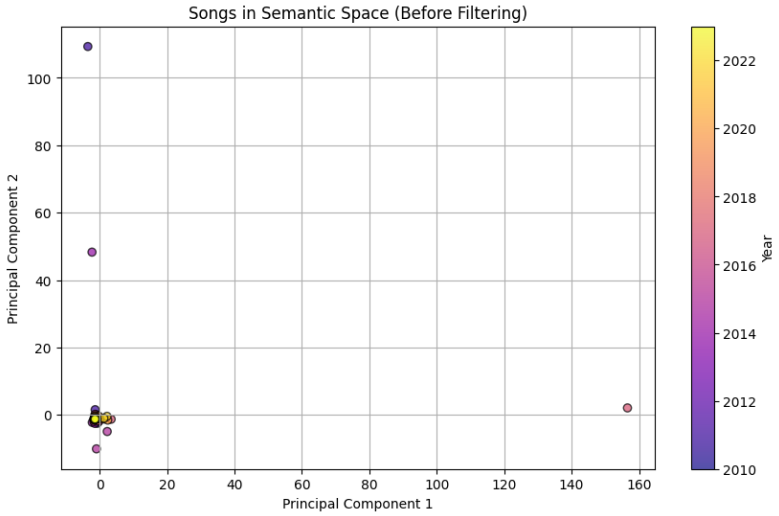
Key Challenges and Solutions:

- Genius API restricted access to lyrics directly → Switched to web scraping AZLyrics.
- Anti-bot detection → Added waiting before sending requests.
- Lack of Metadata storage → Added new fields within Document objects

3. Experiments and Analysis

2D Projection

- Songs plotted in 2D semantic space, colored by year. Dense central cluster observed; some visible outliers.



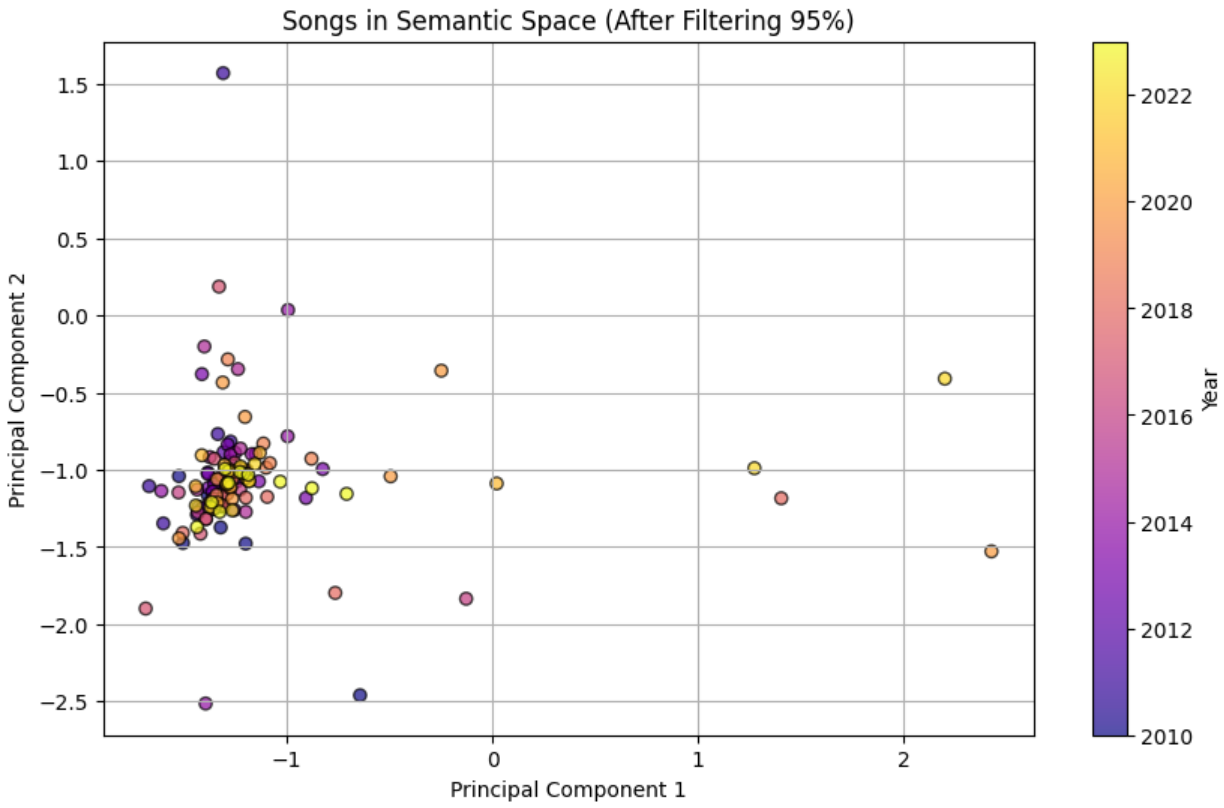
3.2 Outlier Detection

- Outliers identified based on distance from main cluster. Removed the furthest 5% of songs. The Outlier songs removed were the following. Additionally, the most unique words in each song is also printed.

```
 Outlier Songs (Filtered Out):
- 2011 | nickiminaj | superbass | Distance: 109.3463
- 2014 | arianagrande | problem | Distance: 3.0963
- 2014 | meghantrainor | allaboutthatbass | Distance: 48.3492
- 2015 | markronson | uptownfunk | Distance: 5.4100
- 2015 | silento | watchme | Distance: 10.0988
- 2017 | kendricklamar | humble | Distance: 156.5825
- 2017 | migos | badandboujee | Distance: 3.6560

Total outliers removed: 7

 Furthest Outlier:
- 2017 | kendricklamar | humble | Distance: 156.5825
```

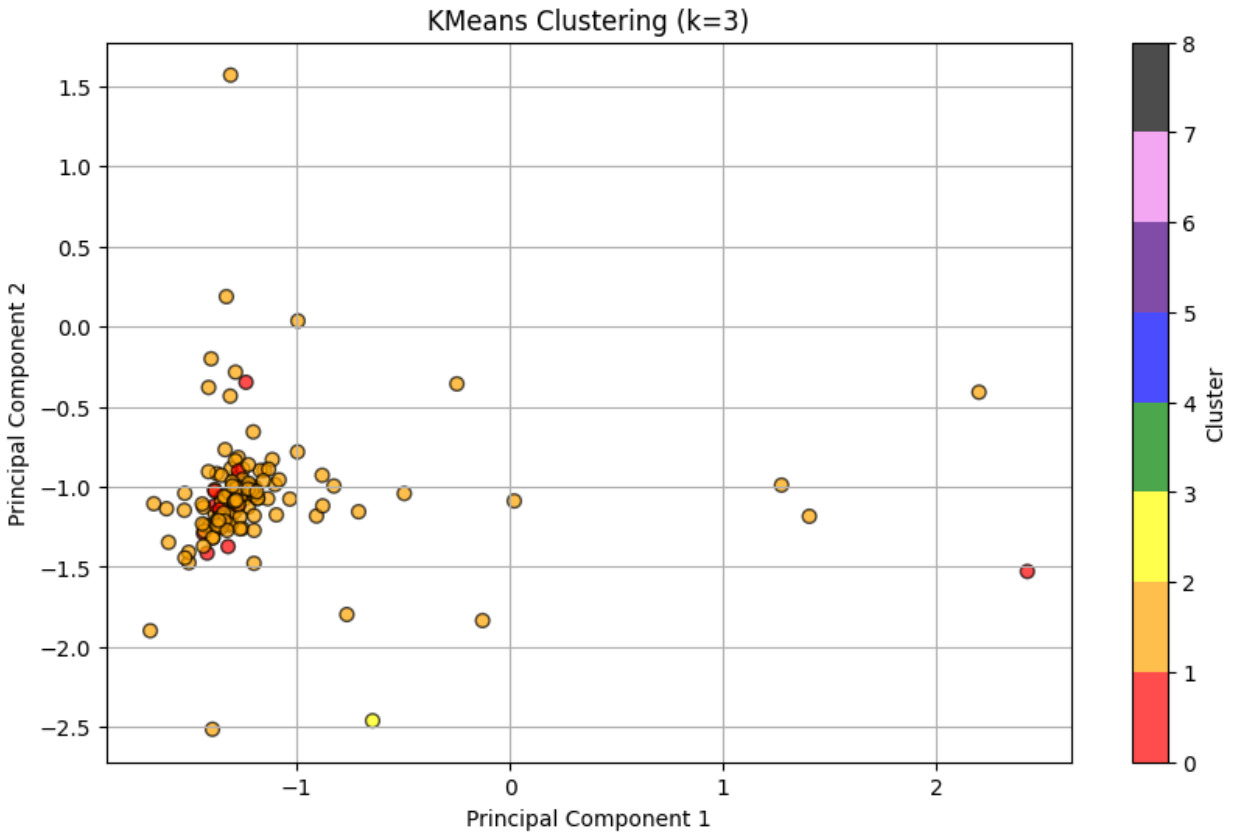
```
 Top words for the most central song:
Year: 2020, Artist: dababy, Song: rockstar
- woo: 14.4114
- nigga: 13.4054
- glock: 12.7848
- cop: 10.9652
- hip: 6.3924
- lamborghini: 6.3924
- pistol: 6.3924
- chop: 6.3924
```

```
 Most distinctive words in Super Bass:
Year: 2011, Artist: nickiminaj, Song: superbass, Distance: 109.3463
- boom: 73.5307
- badoom: 59.6624
- bass: 48.5389
- super: 29.7679
- heartbeat: 9.1464
- drum: 5.4826
- mack: 5.4694
- yes: 5.0756

 Most distinctive words in Humble:
Year: 2017, Artist: kendricklamar, Song: humble, Distance: 153.4111
- hol: 136.3711
- sit: 42.1210
- lil: 32.8918
- bitch: 31.3109
- humble: 29.2975
- aye: 27.7004
- funk: 6.9455
- stroke: 4.2616
```

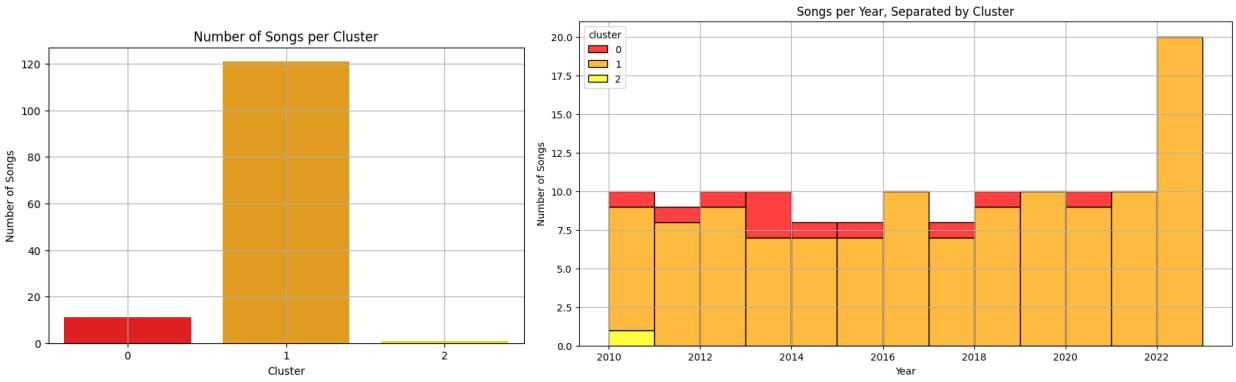


3.3 K-Means Clustering

- Selected k = 3 using the elbow method.
- One large dominant cluster, a significantly smaller cluster and a tiny cluster of 1 song.



- Cluster membership tables by year.



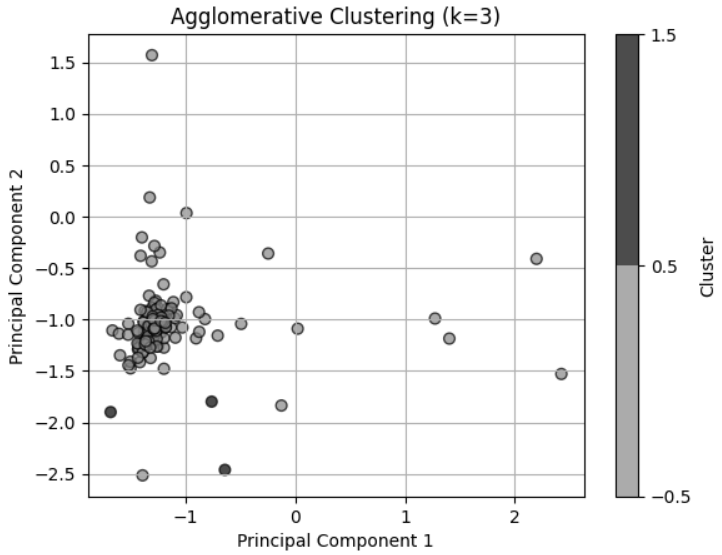
- Top words per cluster and most "central" song (closest to centroid) identified.

```
Top Words per Cluster:

Cluster 0:
- love (0.8223)
- tonight (0.7606)
- like (0.7270)
- yeah (0.7216)
- oh (0.6984)
- youre (0.6948)
- want (0.6612)
- ive (0.6524)
- cant (0.6450)
- ooh (0.6447)

Cluster 1:
- que (26.7539)
- el (16.6059)
- wouh (15.0229)
- tu (14.7608)
- de (14.7608)
- te (11.0706)
- despacito (10.7306)
- quiero (10.7306)
- poquito (8.5845)
- tus (8.5845)

Cluster 2:
- romance (62.2377)
- bad (20.7392)
- want (20.6122)
- revenge (15.0229)
- ra (12.8768)
- roma (12.8768)
- ma (12.8768)
- caught (12.4304)
- gaga (11.0706)
- write (8.2079)
```

```
Top Songs Closest to Cluster Centers:

Cluster 0:
- 2013 | brunomars | wheniwasyourman
- 2022 | adele | easyonme
- 2014 | samsmith | staywithme
- 2014 | johnlegend | allofme
- 2022 | harrystyles | asitwas
- 2015 | wizkhalifa | seeyouagain
- 2010 | ladyantebellum | needyounow
- 2016 | justinbieber | loveyourself
- 2019 | postmalone | sunflower
- 2022 | eltonjohn | coldheartpnauremix

Cluster 1:
- 2018 | cardi-b | ilikeit
- 2017 | luisfonsi | despacitoremix

Cluster 2:
- 2010 | ladygaga | badromance
```

3.4 Agglomerative Clustering

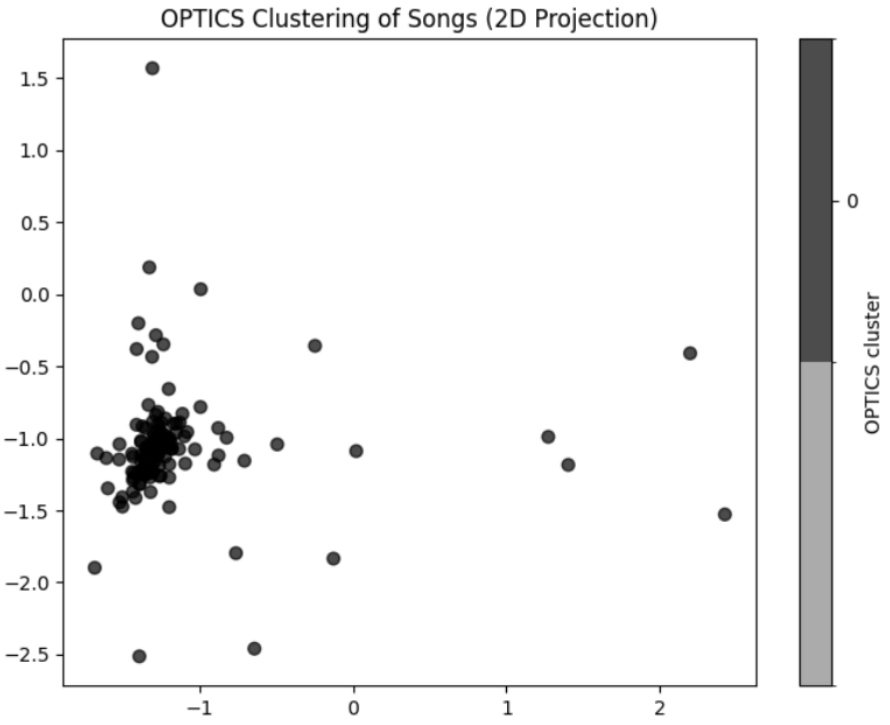
- Produced one dominant large cluster with minor smaller clusters, confirming most pop lyrics are semantically dense and not cleanly hierarchical.



3.5 OPTICS Clustering

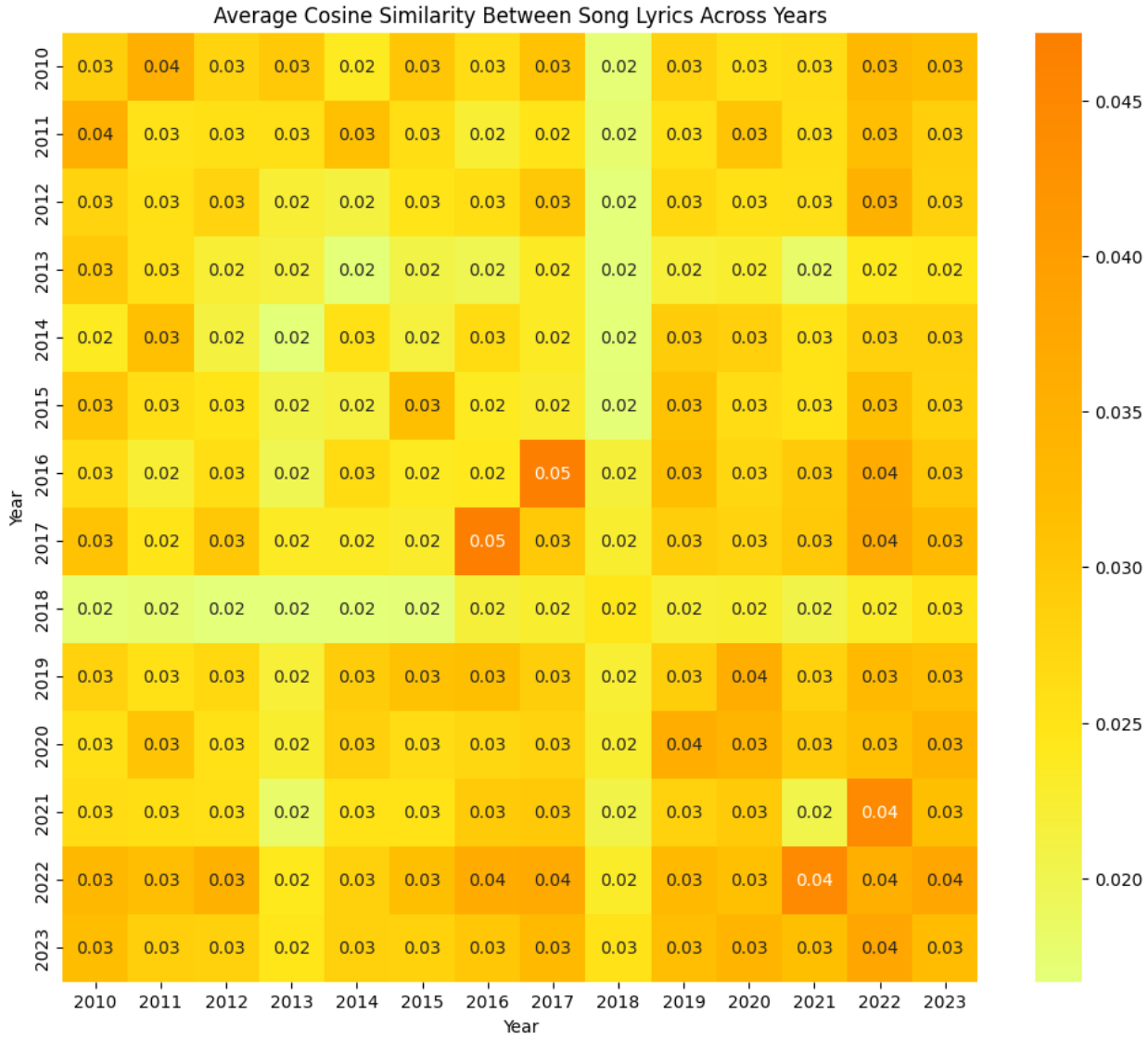
- High proportion of noise detected.
- Few distinct, strong clusters formed, suggesting lack of natural dense groupings.

```
Cluster 0: 133 songs
Noise: 7 songs
```



3.6 Cosine Similarity Heatmap

- Heatmap of average year-to-year similarity.
- Songs modestly similar other songs from surrounding years.
- Songs from 2018 is highly dissimilar to pop music overall, to a lesser degree so are songs from 2013.
- Apart from 2018, songs begin to increase in similarity from 2016 onwards, songs from that time are mutually more similar to each other, compared to songs before 2016.
- Songs from 2010 – 2013 are generally similar to pop songs from this period of analysis overall.



3.7 Most Similar and Most Different Song Pairs

- Identified song pairs with highest and lowest cosine similarity.
- High similarity even for semantically dissimilar songs often due to repetitive, filler-word heavy lyrics, and repetitive lyrical patterns.
- Cosine similarity is 1.0 due to compression of dimensions via SVD

```
Most similar songs:
1. justinbieber – sorry
2. postmalone – congratulations
Cosine similarity: 1.0000
```

4. Discussion

Clustering Behavior:

Songs weakly cluster by lyrical/semantic theme. Year-based clustering is not strongly evident, although are typically. There is a large cluster that appears in almost every clustering algorithm, which suggest that pop music is mostly lyrically homogeneous, with many songs sharing common vocabulary and structure regardless of year or artist.

Lyrical Evolution:

Lyrics content generally maintain the same level of difference/similarity throughout this period, with a few outliers like 2018 and 2013, where lyrics are relatively dissimilar to all other songs of this time. And additionally, the years 2017 and 2016 have really similar songs.

Post-2020 songs are more similar overall, potentially due to viral music trends and streaming platform influences.

Hypotheses Reflection:

H1: Songs from similar years will have higher lyrical similarity.

Partially supported. This varies on a case to case basis, and not necessarily holds true overall. We can see in the KMeans bar chart of songs per year separated by cluster, the songs from each cluster is distributed relatively evenly across the different years. Additionally, in the cosine similarity heat map, there is not a clear relationship between songs within the similar years, as we do not see any sign of increased similarity along to diagonal. However, we can see slightly increase similarity for the songs from years 2016-2024, which potentially suggest some sort

of an era in pop music. The same thing can be said about songs from 2010 – 2012. And the remaining years 2012- 2016, songs from this time are not really similar to each other, which suggest an era of more distinctive experimental music.

H2: Songs will cluster according to their lyrical, semantic meaning.

Disproven. Through this analysis, songs that have widely distinctive meanings, such as Post Malone's *Congratulations* and Justin Bieber's *Sorry*, are determined to be similar, which does not hold for semantic meaning. Additionally, songs like Nicki Minaj's *Super Bass* and Kendrick Lamar's *Humble.* were determined to be outliers, and the further away from all of the other pop songs of the entire 2010-2024 time period. Clearly this is not true as semantically, the message of Nicki Minaj's *Super Bass* contains typical themes of romance, and *Humble.* contains a common theme of ego and success as well. My guess is that this is due to the lyrical structure of the text, as choruses are typically repeated, and uncommon words like "boom" are highlighted disproportionately, just as it's repeated many times in the chorus.

H3: Pop lyrics have become more homogeneous over time.

Mostly Supported. We can see that past 2016, there's been a slight increase in similarity amongst songs from the 2016-2024 period. Whether this is potentially semantic meaning, song structure, or simply the amount of words that do not contribute meaningfully, we can still conclude that pop lyrics have become more homogeneous over time at this point. However, from 2010 to 2014, we actually see in the heap map, the similarity of songs falling, but since then, we have seen that this claim holds.

Limitations:

Dimensionality reduction with the SVD may compress meaningful variance contained within the text. Potentially compressing different words/meanings together. Additionally, analysis purely using TF-IDF vectors does not fully maintain semantic context, instead focusing on pure pattern matching in terms of similar patterns within the lyrical structure, for example repeated words that don't carry distinctive meaning. Additionally, analysis with a larger data set of lyrics and a larger span of years could be interesting to draw potential more conclusive data.

Future Directions:

Develop some sort of protocol for preprocessing to systematically remove "lyric stopwords" which I define to be words that are vocalizations that does not meaningfully contribute to the message of a text. For example, these are terms like "mmm," "ooh," "yeah," "ayy."

Reduce dimension compression through the SVD, and see if that would improve semantic accuracy of this form of analysis.

5. Conclusion

Pop lyrics from 2010 to 2024 show minimal thematic evolution with increased homogenization in recent years. In particular, semantic analysis via clustering failed to capture the semantic meaning of lyrics as it was dominated by noise. Clustering based solely on TF-IDF embeddings reveals limited separation between songs, suggesting that lexical patterns primary dominate pop music lyrics.

Appendix

1. Code: Java code submitted separately, Google Colab link: https://colab.research.google.com/drive/13vJBq8wTMrBLFjapeaVnfeHeJkm3Ouc8?usp=sharing
2. Data from web scraping sources: Wikipedia.com, azlyrics.com
3. NTLK's list of stop words: https://gist.github.com/sebleier/554280