Ay-Yo! User Similarity

Marti Heit and Emma Rainer

Project Goals

- Create similarity metric for Ay-Yo! Users based on their taste in music
  - Could be used to recommend similar users
- Explore ways to solve the cold start problem
  - Ay-Yo! is a small, new social media, so we don’t have the amount of data necessary to get good results from traditional deep learning techniques (natural language processing/matrix factorization)
  - Instead, can we leverage transfer learning to gain knowledge about a user’s taste in music to recommend similar users?

Our Process

1. Connected to DynamoDB using boto3 to pull Ay-Yo! user data, including the songs posted by each user
2. Connected to the Genius API to pull song lyrics for every post an Ay-Yo! user has made
3. Used various tokenization methods (word tokenization, subword tokenization) to prepare song lyrics for embeddings
4. Used various embedding techniques to create vector representations of each song posted on Ay-Yo!
5. Averaged vector representations of songs posted by each user to create representation for each user
6. Computed Euclidean distance between users to determine similarity

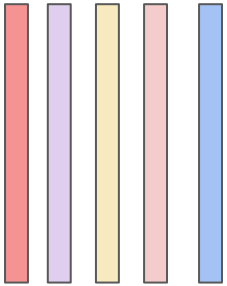
The Architecture

Lyrics

tokenizer

Car
Rides
Malibu
Strawberry
Ice
Cream

embedding

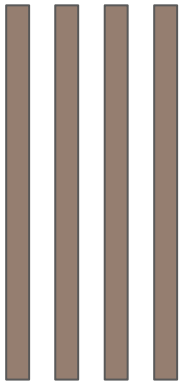


Vector embedding of words/tokens

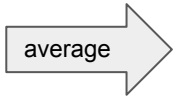
average

Representation of a song

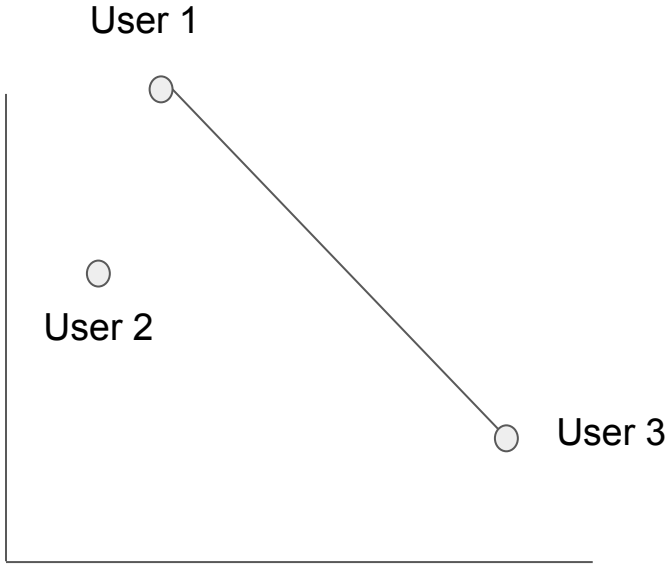
The Architecture (cont)



Vector representation of every song you have posted



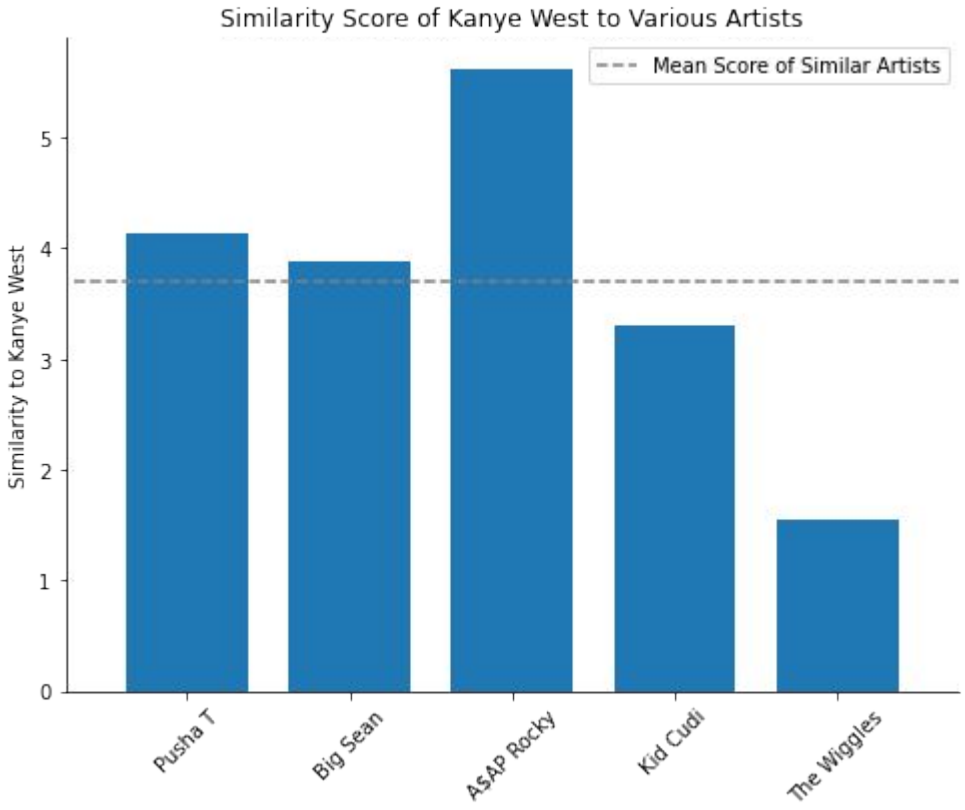
Vector representation of Ay-Yo! user

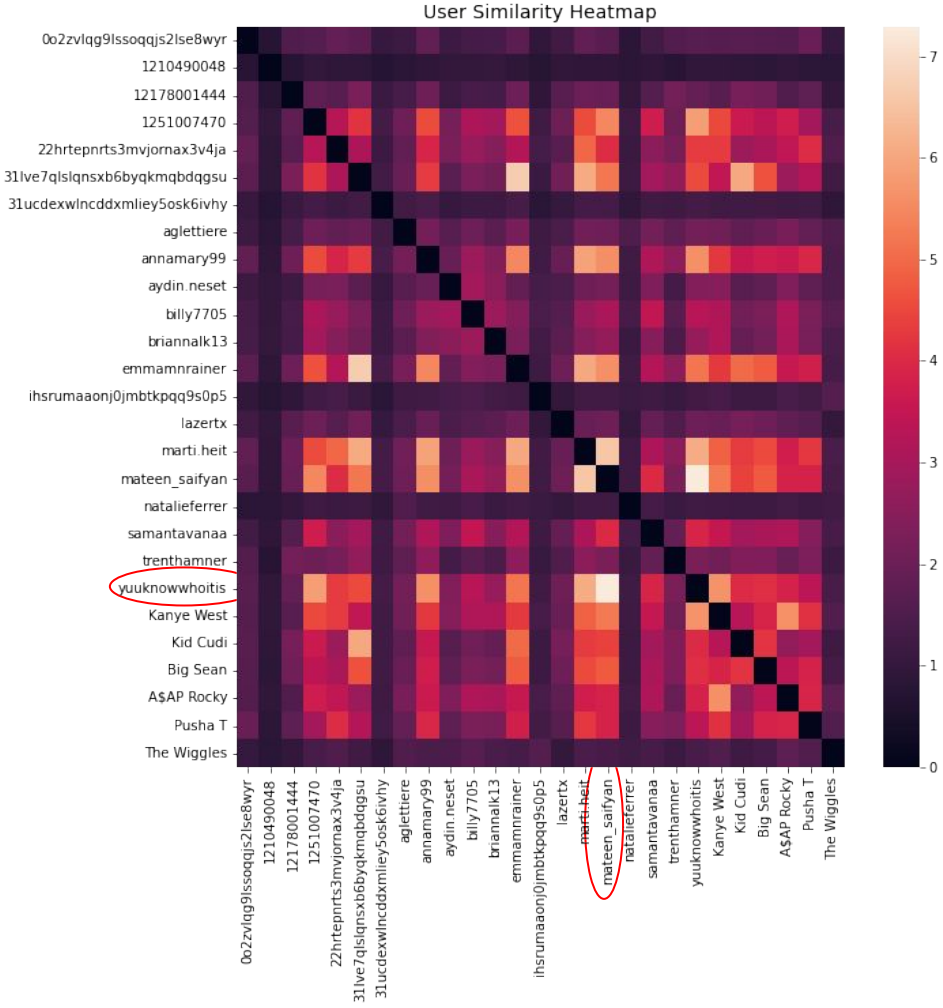


$$\text{similarity score} = \frac{1}{\|\vec{u}_i - \vec{u}_j\|}$$

Validation

- Clearly, our first task (creating similarity scores) is unsupervised
  - We don’t know which Ay-Yo! users are actually most similar to each other, and there is no way of accurately creating these labels
- To test our algorithm, we created representations for artists, then computed their similarity scores to the Spotify recommended similar artists
- We computed their similarity score to a artist who we could reasonably assume is not similar to the artist


Similarity Score of Kanye West to Various Artists
Mean Score of Similar Artists
Similarity to Kanye West
0
1
2
3
4
5
Pusha T
Big Sean
ASAP Rocky
Kid Cudi
The Wiggles


User Similarity Heatmap
0o2zvlqg9lssoqqjs2lse8wyr
1210490048
12178001444
1251007470
22hrtepnrts3mvjornax3v4ja
31lve7qlslqnsxb6byqkmqbdqgsu
31ucdexwlncddxmliey5osk6ivhy
aglettiere
annamary99
aydin.neset
billy7705
briannalk13
emmamnrainer
ihsrumaaonj0jmbtkpqq9s0p5
lazertx
marti.heit
mateen_saifyan
natalieferrer
samantavanaa
trenthamner
yuuknowwhoitis
Kanye West
Kid Cudi
Big Sean
A$AP Rocky
Pusha T
The Wiggles
0
1
2
3
4
5
6
7

What Worked

- Preprocessing
  - GloVe Embeddings (Wikipedia)
  - Tokenization (BERT subword tokenizer trained on Wikipedia corpus)
- Analysis
  - Ranking users based on similarity scores
  - Generally distinguishing similar artists from non-similar artists

```
get_most_similar_users('marti.heit', dataset)[1:5]
```

```
[('mateen_saifyan', 6.5895859605959455),
 ('yuuknowwhoitis', 6.1180410098315345),
 ('31lve7qlslqnsxb6byqkmqbdqgsu', 6.087149022660887),
 ('emmamnrainer', 6.030698942969794)]
```

```
get_most_similar_users('marti.heit', dataset)[-1]
```

```
('1210490048', 0.9182611924401537)
```

What Didn’t Work

- One of our blockers in this project was the amount of data we had
  - We only had 20 Ay-Yo! users who have posted between 1-30 songs
- Our algorithm had some trouble distinguishing subtleties in how dissimilar users are
- Lyrical analysis doesn’t account for differences in beat, rhythm, etc
- Our lyrics were fairly messy, perhaps finding and removing different stop tokens could have improved our results
- We can’t really apply this at this point, since almost all our Ay-Yo! users are already following each other

```
user1, centroid1 = dataset[user2idx['Kanye West']]
user2, centroid2 = dataset[user2idx['The Wiggles']]
print(f"Similarity score between {user1} and {user2} is {similarity_score(centroid1, centroid2)}")
```

```
Similarity score between Kanye West and The Wiggles is 1.5429937150739725
```

Future Extensions

- Explore applying user music taste representations as a feature in a larger matrix factorization model to recommend similar users
- Apply similarity scores to rank relevant results when searching for users/songs