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?
- Explore making music recommendations based partially on word embeddings of user Ay-Yo! histories

Task 1: Create vector representations of Ay-Yo! user app histories and compute similarity metric

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 (**found the centroid**) of songs posted by each user to create representation for each user
6. Computed **Euclidean distance** between users to determine similarity

The Architecture


Deja Vu

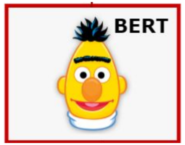
Car rides to Malibu
Strawberry ice cream, one spoon for two
And trading jackets
Laughing 'bout how small it looks on you (Ha-ha-ha-ha)
Watching reruns of Glee
Bein' annoying, singing in harmony
I bet she's braggin' to all her friends
Saying you're so unique (Huh)
So when you're gonna tell her
That we did that too, she think it's special?
But it's all re-used
That was our place, I found it first
I made the jokes you tell to her
When she's with you

Do you get deja vu when she's with you?
Do you get deja vu? (Huh)
Do you get deja vu, huh?

What do you call her? Almost say my name
'Cause let's be honest, they kinda do sound the same
Another actress, I hate to think that I was just your type
And I bet that she knows Billy Joel
'Cause you played her uptown Girl
You're singing it together
Now I bet you even tell her how you love her
In between the chorus and the verse
So when you're gonna tell her
That we did that too, she think it's special?
But it's all re-used
That was our place, I found it first
I made the jokes you tell to her
When she's with you

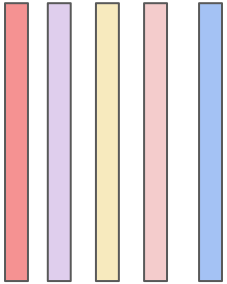
Lyrics

tokenizer



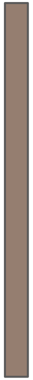
Car
Rides
Malibu
Strawberry
Ice
Cream

embedding



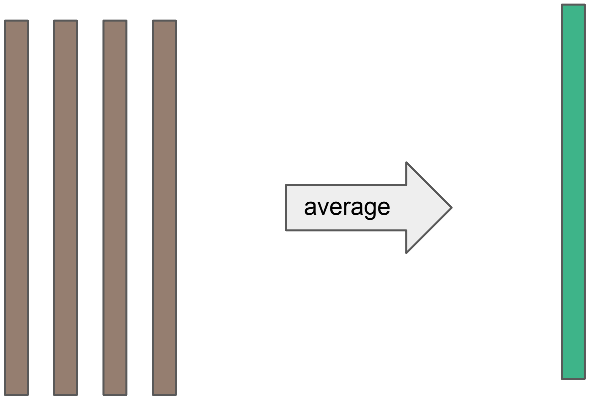
Vector embedding of words/tokens

average



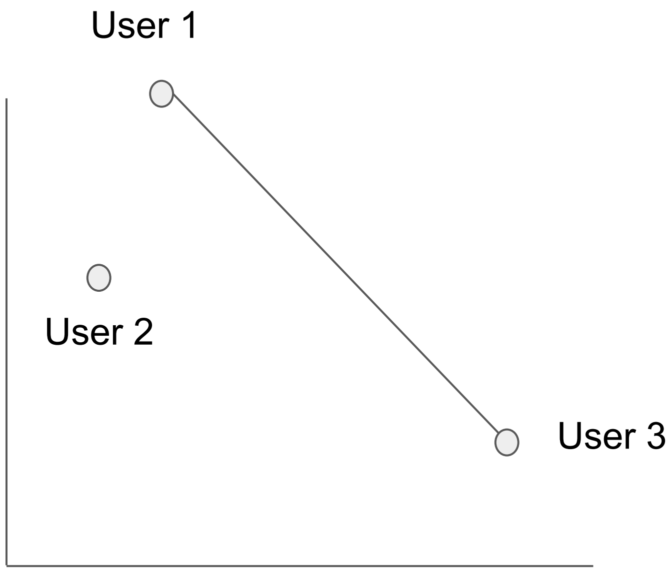
Representation of a song

The Architecture (cont)



Vector representation of every song you have posted

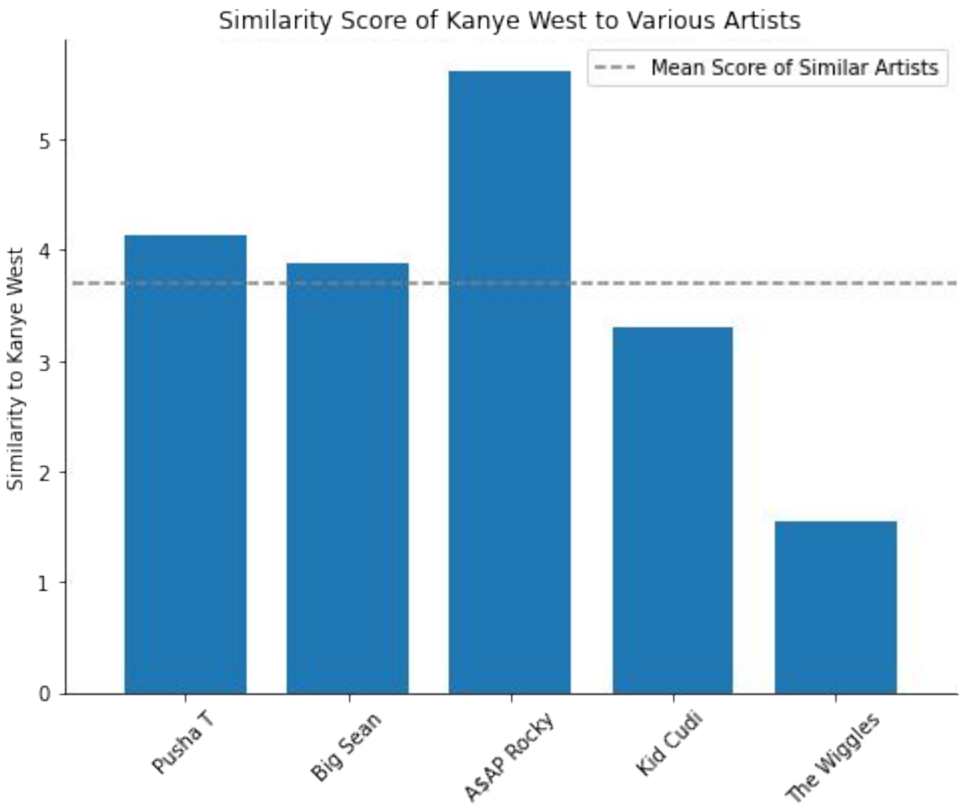
Vector representation of Ay-Yo! user

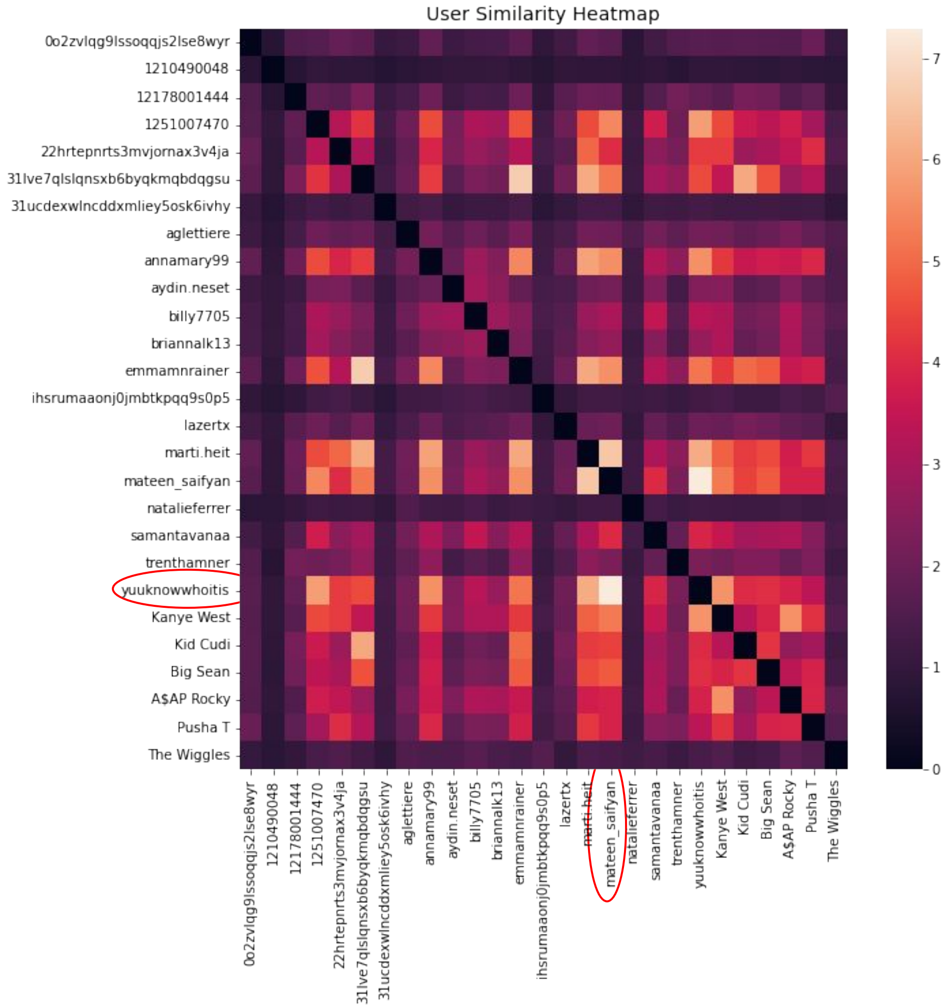


$$\text{similarity score} = \frac{1}{\|\vec{u}_i - \vec{u}_j\|}$$

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 well known 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
```

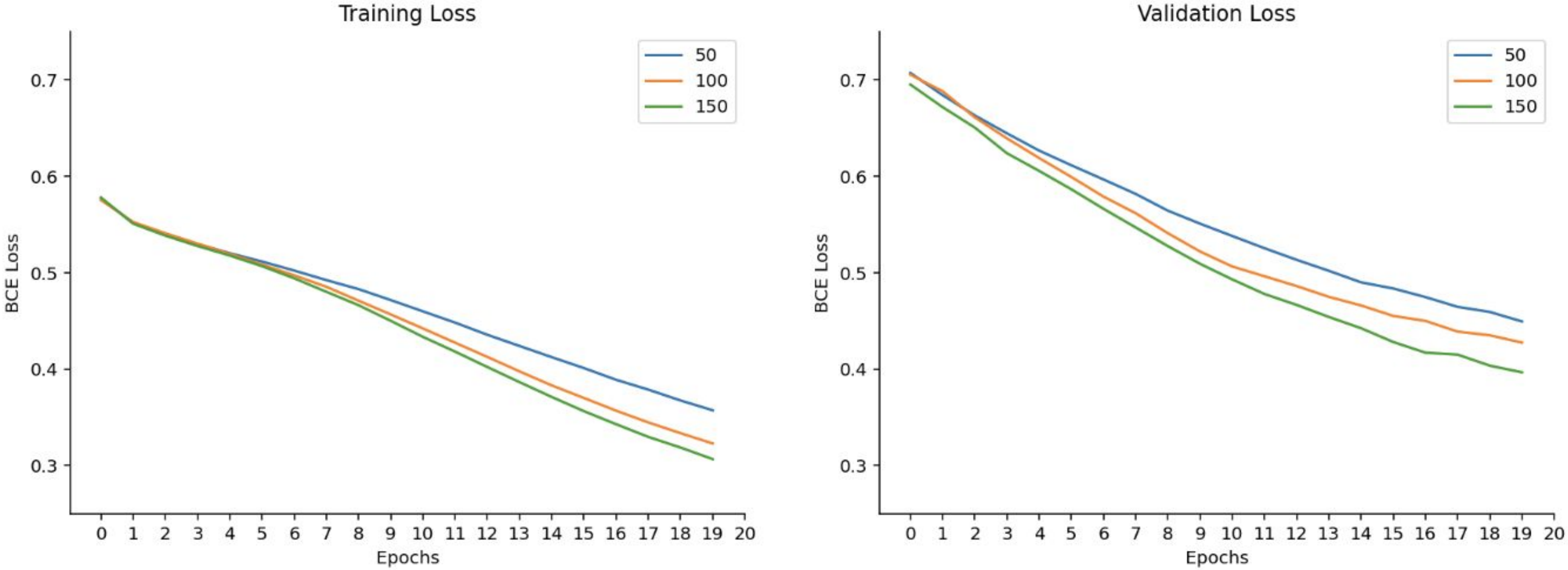
Task 2: Use vector representations as features to predict implicit ratings

Our Process

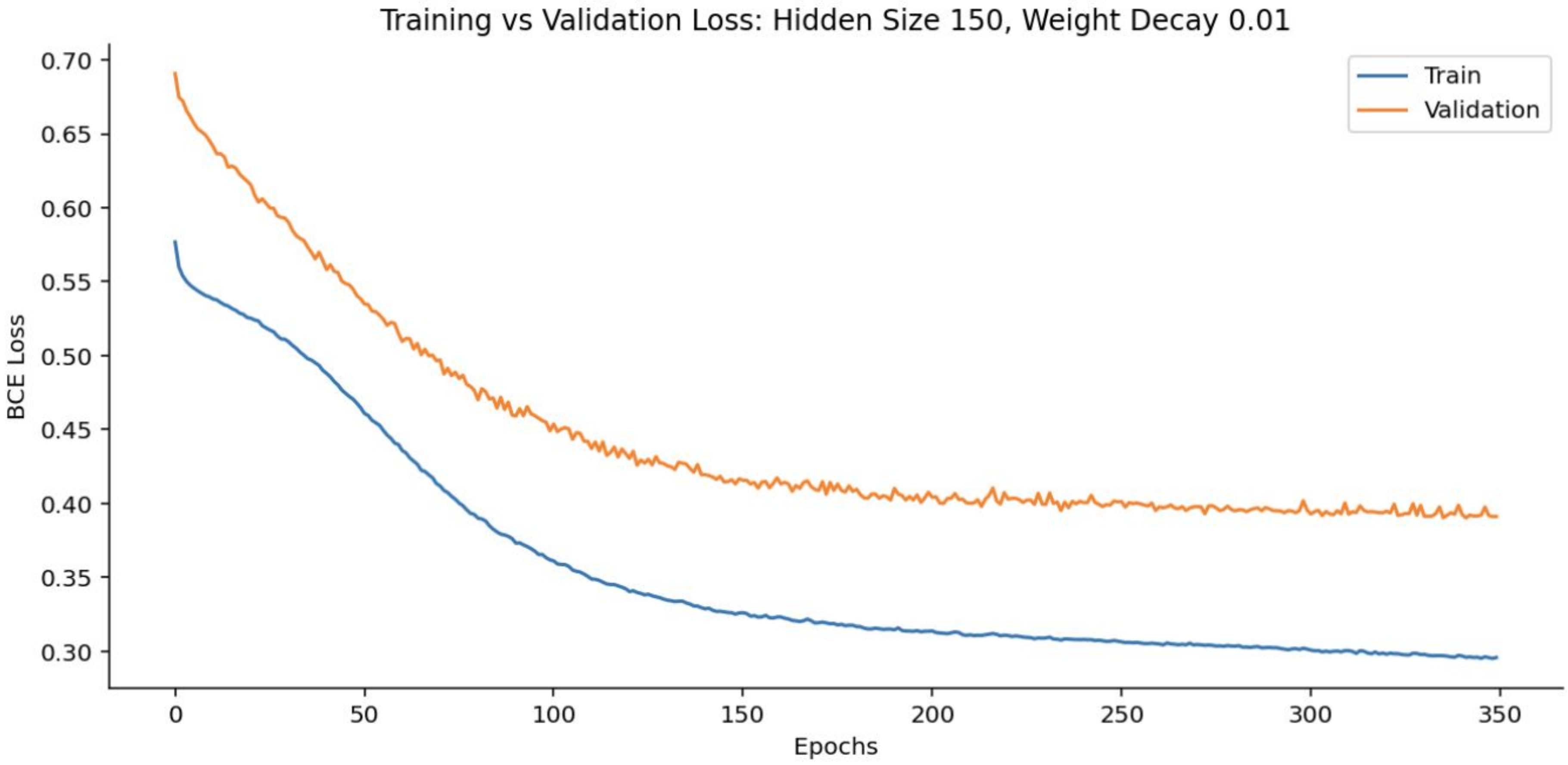
1. Retrieve additional song features using the **Spotify API**
   a. Energy, loudness, danceability, speechiness
2. Assign each Ay-Yo! user and their posts to the positive class (“1”)
3. Generate **negative samples** by leveraging our similarity metric
   a. Assign each Ay-Yo! User and the posts of their least similar Ay-Yo! Counterpart to the negative class (“0”)
4. For each user/song combination: we have information about the song from Spotify, the song’s lyrical embedding, and the user’s centroid (vector representation)
5. Combine all the data to **predict implicit ratings** (1/0) using a 2-layer vanilla neural network
6. **Hyperparameter tune** the hidden layer size using validation metrics
7. Applied **weight decay** as a regularization technique

Results

- **Baseline model metrics**:
    - Training Loss: 0.6995
    - Validation Loss : 0.6943
- **Optimal value found for hidden size parameter:** 150
    - With more time and data, it would be beneficial to do an extensive grid search rather than just a few values
- **Best Model Performance**
    - Training Loss: 0.2950
    - Validation Loss : 0.3901
- **Improvements:**
    - 57.1532% decrease in training loss from baseline
    - 43.8139% decrease in validation loss from baseline
- **Generalizability** needs improvement
    - Weight decay worked in reducing validation loss to an extent, then validation loss began to plateau at around 200 epochs



Training Loss
Validation Loss
50
100
150
BCE Loss
Epochs
0.3
0.4
0.5
0.6
0.7
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


Training vs Validation Loss: Hidden Size 150, Weight Decay 0.01
Train
Validation
BCE Loss
Epochs
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
50
100
150
200
250
300
350

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