



Jester Jokes Recommender

Recommends daily jokes to our subscriber

By Susan Qu

Outline

- The Business Problem
- Data Gathering & Dataset Description
- Modeling Approaches Overview
- Data Cleaning/Transformation
- EDA
- Word Embedding and Topic Modeling with NMF
- Recommendation System Approaches
- Comparison of Performance
- Improvement Ideas

The Business Problem

We have more than 36K users subscribed to our online joke service, each day we want to send a new joke to the user via email, that he hasn't rated before. In this project, we are looking to find the most suitable algorithm to find and recommend the relevant jokes to users.



Data Gathering & Dataset Description

Data Gathering

The original Jester jokes dataset was made available by Ken Goldberg at UC Berkeley AUTOLab, for the Eigentaste: A Constant Time Collaborative Filtering Algorithm. The first dataset (Dataset 1) had 4.1 million ratings, values from (-10.00 to +10.00) of 100 jokes from 73,421 users: collected between April 1999 - May 2003. We will be using a subset of the data for the current project.

Dataset Description

- JokeText.csv contains the Id of the joke and the complete joke string.
- UserRatings1.csv contains the ratings provided by the first 36710 users.
- The dataset is arranged such that the initial users have rated higher number of jokes than the later users.
- The rating is a real value between -10.0 and +10.0.
- The empty values indicate that the user has not provided any rating for that particular joke.



Modeling Approach

Since there are no categorical data for each joke, to establish item-item similarity, we will use **topic modeling method via NMF** to extract topics from each joke, and create **cosine similarity matrix** based on the topics to find jokes that are similar to each other.

This **content based similarity matrix** will then be used in the recommendation system, to recommend our users the similar jokes if user hasn't rated this joke already.

Another recommender system we explore here is the **user-user similarity collaborative filtering** approach, where we establish similarity matrix based on similar users' ratings of jokes in query.

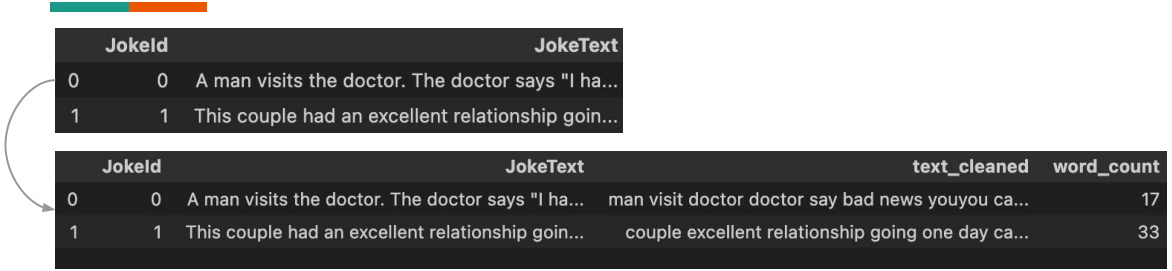
To evaluate the performance of each model, we will split the dataset, into train and test set. Since not all user have rated all the jokes, we will reserve 20% of rated jokes from each user as test set. We'll establish baseline by using the user's average rating and use **RMSE as the metric to evaluate the models**.



Modeling Step

- EDA
- Topic Modelling via Unsupervised Algo (NMF)
 - Joke text embedding vectorization
 - Jokes-Topic Distribution Matrix
 - Establish item to item similarity based on n topics
- Recommender
 - Split the current user data into train and test
 - Baseline approach using user's average
 - Content similarity approach
 - User to User Collaborative filtering approach
- Apply Predict function to compare the approaches

Data Cleaning and Transformation



A diagram illustrating data transformation. It shows a curved arrow pointing from a simple table with two columns to a more complex table with four columns. The simple table has columns 'JokeId' and 'JokeText'. The complex table has columns 'JokeId', 'JokeText', 'text_cleaned', and 'word_count'.

	JokeId	JokeText
0	0	A man visits the doctor. The doctor says "I ha...
1	1	This couple had an excellent relationship goin...

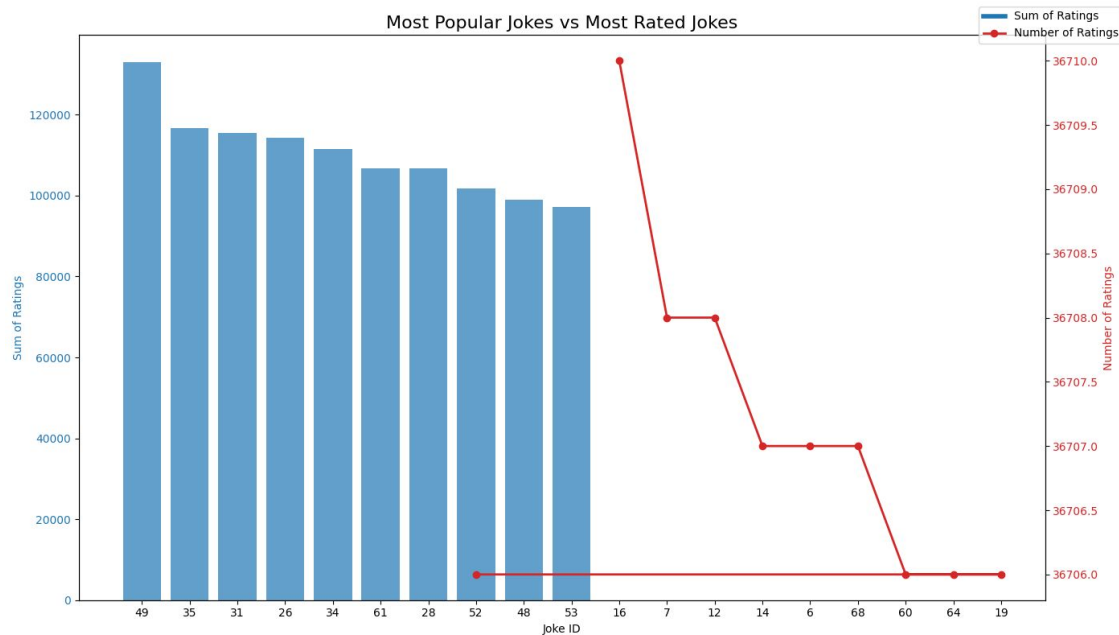
	JokeId	JokeText	text_cleaned	word_count
0	0	A man visits the doctor. The doctor says "I ha...	man visit doctor doctor say bad news youyou ca...	17
1	1	This couple had an excellent relationship goin...	couple excellent relationship going one day ca...	33

	JokeId	User1	User2	User3	User4	User5	User6	User7	User8	User9	...	User36701	User36702	User36703
0	0	5.1	-8.79	-3.50	7.14	-8.79	9.22	-4.03	3.11	-3.64	...	NaN	NaN	NaN
1	1	4.9	-0.87	-2.91	-3.88	-0.58	9.37	-1.55	0.92	-3.35	...	NaN	NaN	NaN

Total number of ratings: 3012090

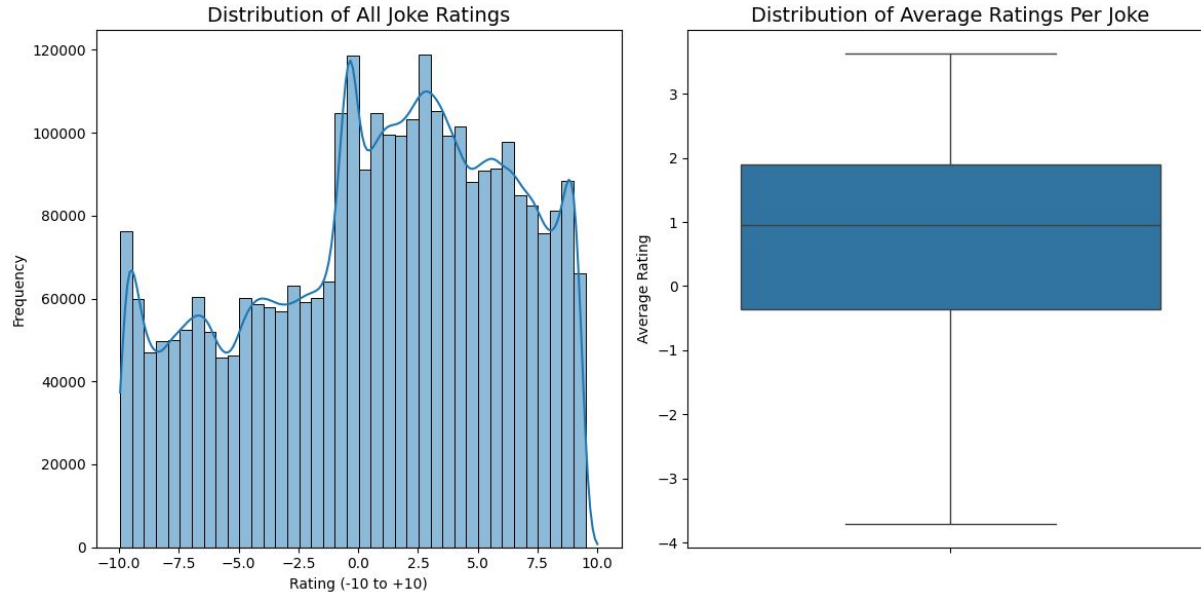
	JokeId	UserID	Rating
0	0	User1	5.10
1	1	User1	4.90
2	2	User1	1.75
3	3	User1	-4.17
4	4	User1	5.15

EDA - most popular vs most rated



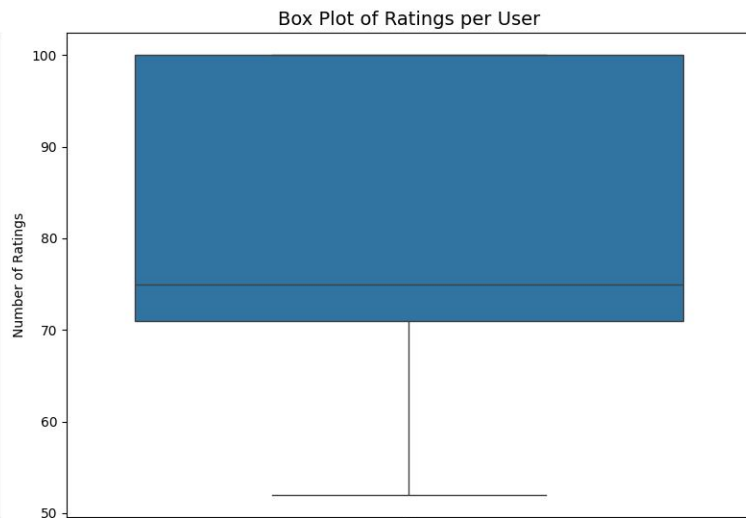
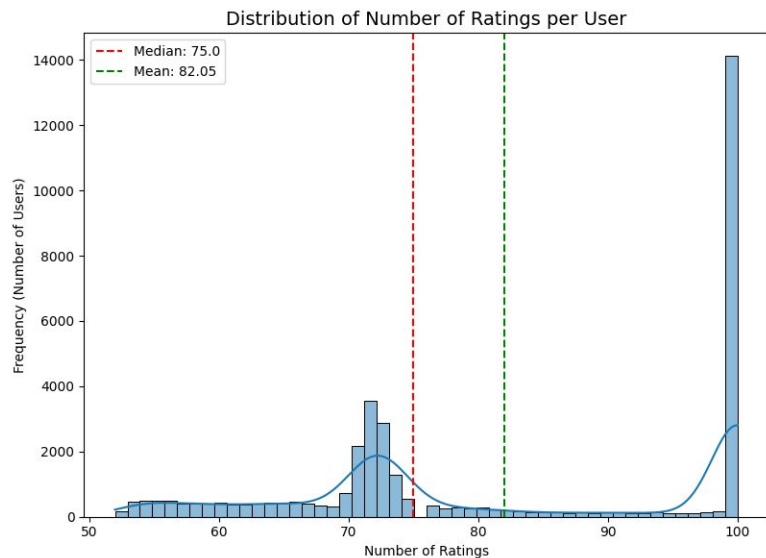
The most popular jokes might not be the most rated jokes

EDA - Distribution of Ratings



- More jokes are rated positively than negatively, looks to be 50% more by looking at the histogram. Large concentration of rating are between 1-6.
- Half of the jokes are rated 1 and above, half are rated 1 and below.

EDA - Distribution of Num of Ratings per User



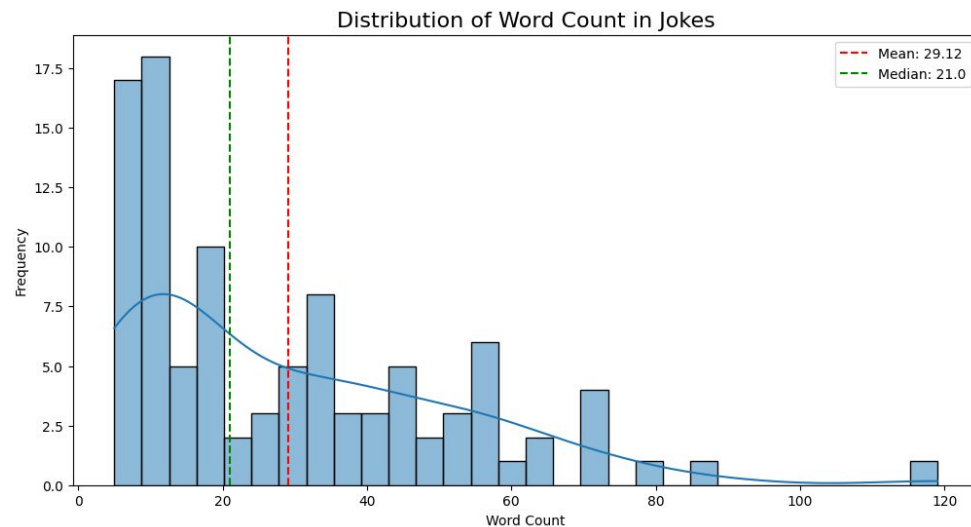
Total number of users: 36710

- Average num of ratings per user: 82.05
- Median num of ratings per user: 75.0
- Min num of ratings ratings per user: 52
- Max num of ratings per user: 100

Percentiles:

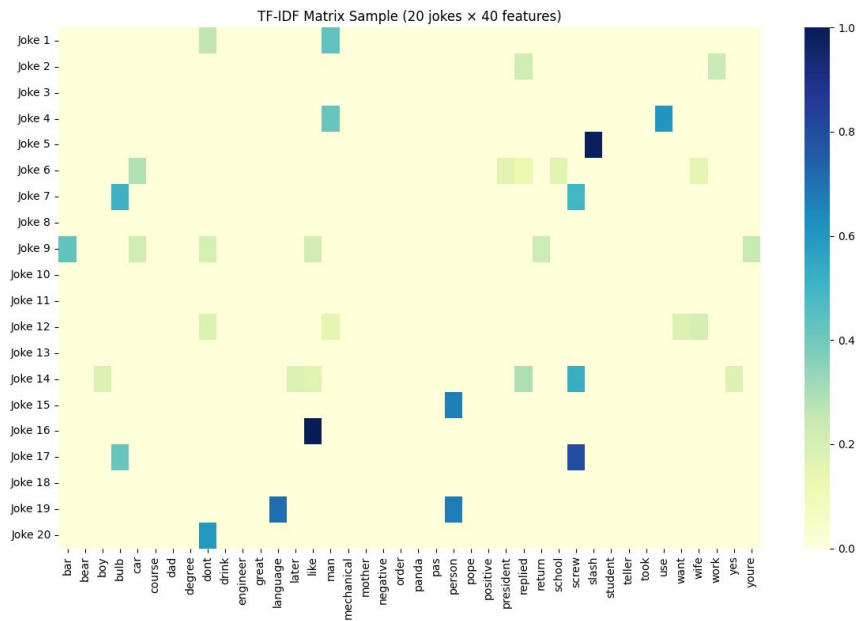
- 25th percentile: 71.0
- 75th percentile: 100.0
- 90th percentile: 100.0
- 95th percentile: 100.0

EDA - Distribution of Word Count



- The avg joke has 29 words.
- The longest joke has about 118 words.
- We only to use 119 as the ``max_features`` in the vectorization

Word Embedding
TF-IDF Matrix Sample (20 jokes \times 40 features)



NMF: Jokes-Topic Distribution Matrix (20 topics)

We want to establish joke to joke similarity matrix based on the jokes-topic distribution.

Ex

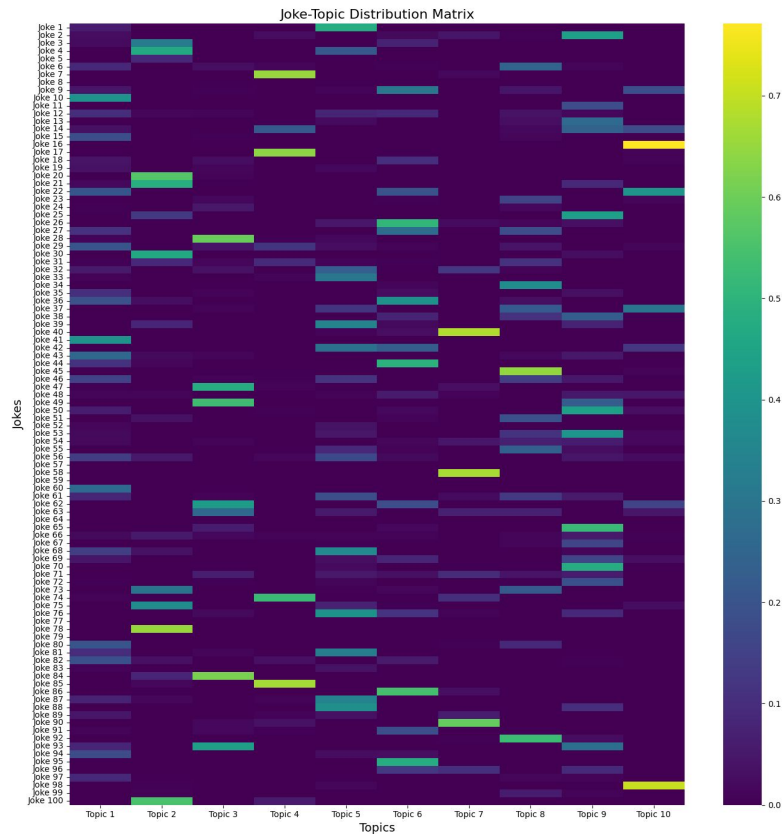
joke 6 might be similar to joke 16

Joke 16 text: How many feminists does it take to screw in a light bulb?

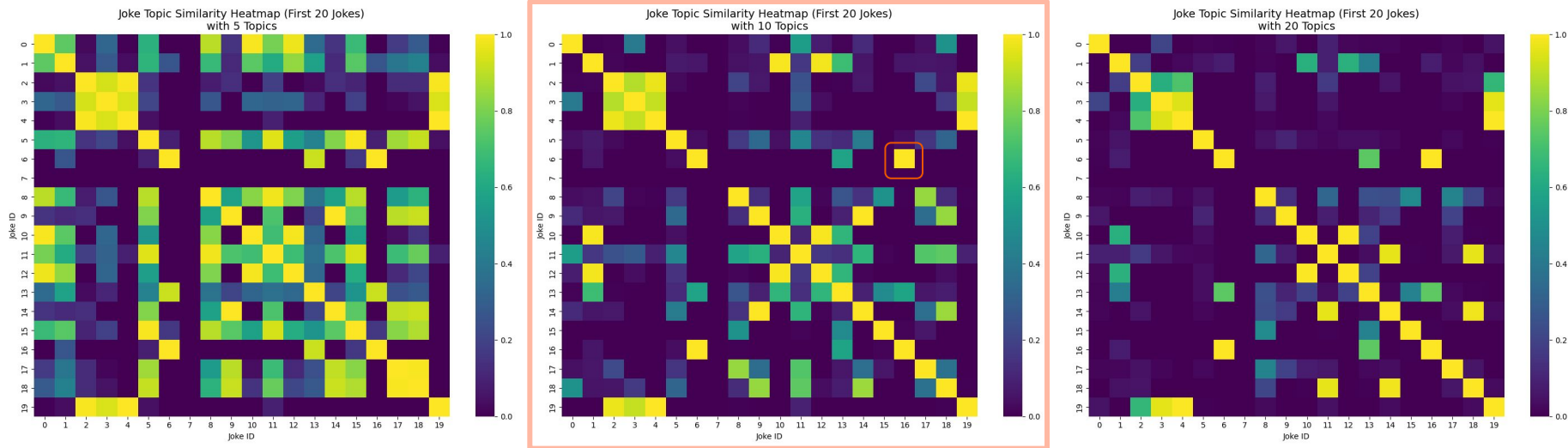
... ..

Joke 6 text: How many men does it take to screw in a light bulb?

... ..



Joke Topic Similarity Heatmap Comparison (First 20 Jokes)



- With more topics, the less commonality between jokes that can be found.
- With less topics, there are more jokes that are similar to each other.

Unlike LLM, NMF is unable to capture underlining contextual meaning of the jokes

- We might want to use a slightly higher number of topics, without running out of similar jokes
- Will choose **10 topics** for now


User-User Similarity Score Matrix

user-to-user similarity matrix using cosine similarity based on joke ratings.

```
User similarity matrix shape: (36710, 36710)
User similarity matrix (first 5 rows and columns):
[[ 1.          -0.16951129  0.0920009  0.24133247  0.36998924]
 [-0.16951129  1.          -0.18724276  0.03312991 -0.08008281]
 [ 0.0920009  -0.18724276  1.          0.02270996  0.30826777]
 [ 0.24133247  0.03312991  0.02270996  1.          0.25929823]
 [ 0.36998924 -0.08008281  0.30826777  0.25929823  1.          ]]
```

Find similar users to a particular user

```
Top 10 similar users to User36704:
User: User36583, Similarity Score: 0.6401
User: User36404, Similarity Score: 0.6119
User: User35928, Similarity Score: 0.5961
User: User36095, Similarity Score: 0.5869
User: User31076, Similarity Score: 0.5861
User: User36658, Similarity Score: 0.5855
User: User36416, Similarity Score: 0.5822
User: User35800, Similarity Score: 0.5751
User: User35057, Similarity Score: 0.5719
User: User35944, Similarity Score: 0.5657
```



Recommendation Systems

- Baseline approach
 - predict the test rating by user's avg from the train set
- Content similarity approach
- User to User Collaborative approach
- Apply Predict function to compare the 4 approaches

User Avg Baseline

It serves as a baseline. For each user, it:

- Calculates the average rating that user has given across all jokes in the training set
- Uses this personal average as the prediction for any joke the user hasn't rated yet
- Falls back to the global average rating (across all users) when a user has no ratings in the training set

The method is based on the observation that users tend to have their own "baseline" level of enjoyment or rating style, which can be a surprisingly strong predictor.



Content Based Filtering

This approach leverages the content similarity between jokes to make predictions:

- Uses the joke similarity matrix calculated via NMF topic modeling of joke text
- For each user-joke pair to predict:
 - Finds jokes this user has already rated in the training set
 - Identifies the most similar jokes to the target joke (based on topic similarity)
 - Predicts the rating by taking the average of how the user rated those similar jokes
 - Can use either simple averaging or similarity-weighted averaging of ratings

This approach assumes that if a user liked joke A, and joke B is very similar in content to joke A, they will probably rate joke B similarly.

User-Based Collaborative Filtering

This approach uses the similarity between users to make predictions:

- Uses the precomputed user similarity matrix (based on their rating patterns)
- For each user-joke pair to predict:
 - Finds users most similar to the target user
 - From those similar users, identifies which ones have rated the target joke
 - Predicts the rating based on how these similar users rated the joke
 - Can use either simple averaging or similarity-weighted averaging

This method assumes that users with similar taste patterns will rate jokes similarly, even if the jokes themselves are not inherently similar in content.

Summary of methods and performance

Method	RMSE
Baseline, $\hat{Y}_p = \mu_u$	4.6274
Content based, item-item NMF 10 topics, top 5 sim avg	4.9823
Content based, item-item NMF 10 topics, top 5 sim weighted avg	4.9709
Content based, item-item NMF 10 topics, top 20 sim avg	4.6874
Content based, item-item NMF 10 topics, top 20 sim weighted avg	4.7112
Content based, item-item NMF 20 topics, top 20 sim avg	4.6765
Collaborative filtering, user-user, cosine sim, avg rating of top 10 sim users (300 test user subset)	4.8290
Collaborative filtering, user-user, cosine sim, avg rating of top 50 sim users (300 test user subset)	4.9193
Collaborative filtering, user-user, cosine sim, avg rating of top 5 sim users (300 test user subset)	4.8215

Content Based vs User Based Collaborative Filtering vs User Average Baseline

1. Rating Scale Subjectivity

Users rate jokes on their own subjective scales - one person's "7" might be another person's "5" for the same level of enjoyment. Your baseline accounts for this by simply learning each user's average rating tendency, which can be a powerful signal.

2. Limited Content Signal from NMF

The joke topic modeling using NMF might not be capturing the true elements that make jokes similar in terms of user preferences. Humor is highly contextual and often relies on subtle linguistic features that simple topic models miss.

3. Preference Consistency

For jokes, a user's average rating is often a strong predictor because people tend to have fairly consistent reactions to humor. The baseline captures this pattern directly.

Possible Improvements Ideas

1. Try normalizing ratings by user means before similarity calculations
2. Try embedding-based approaches for better joke representation, such as using BertTopic. It would better capture the semantic meaning the jokes.
3. Employing a deep learning model to perform collaborative filtering.

Thanks!

Github Repo Link
