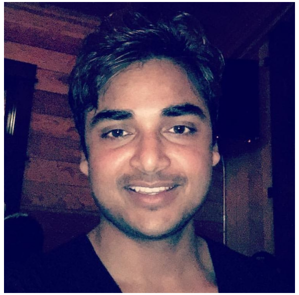
Software
Product Sprint

Team 85

LOL App

Software
Product Sprint

Who We Are?

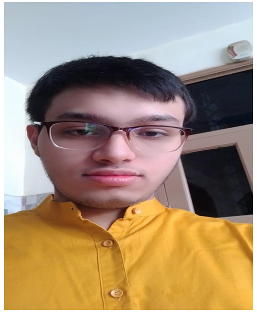


Nityendra
- The PA

Poorvi

Riya

Satya



Shubham

Table of Contents

- Overview
- Motivation
- Demo on UI
- Frontend, Backend, System Architecture
- Demo on Ranking
- Modified Hot Ranking Algorithm
- Collaborative Filtering
- Learnings
- Future Work

Overview and Motivation

Overview

- An android app for sharing jokes and memes
- Top trending memes/jokes based on user interests will appear once user opens the app
- Users can give reactions to the meme/joke: rofl, like, dislike
- Users will have option to upload meme/joke and mention the related topics
- Many more features

Motivation

- Various frontend features: Implementation of UI design in **Flutter**
- Backend: Implementation of API calls in **Node.js,** integration of **Cloud SQL** server with **App Engine** to deploy the application
- **ML** and **Ranking** algorithms

Something to learn for each team member while building from scratch and also fun working with memes :p

DEMO on UI

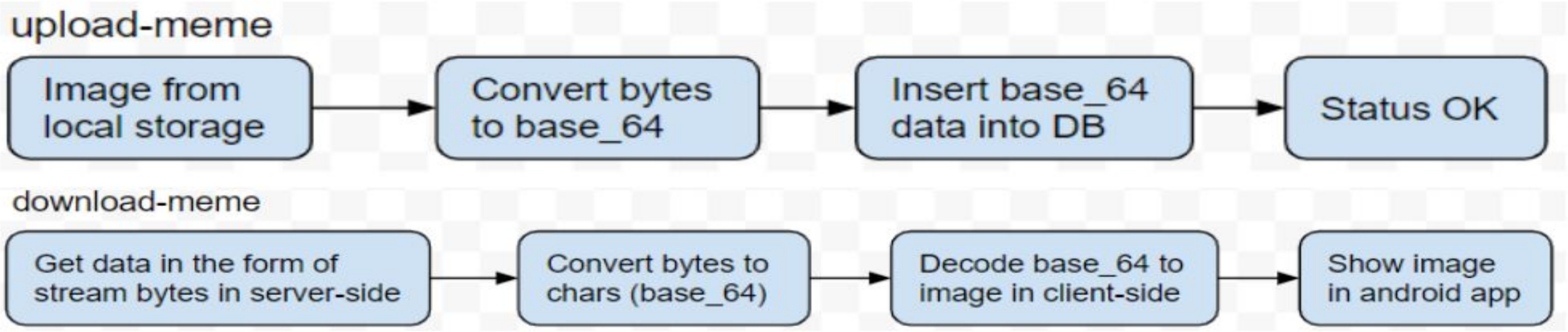
Frontend, Backend, System Architecture

Features

- signup
- signin
- signout
- upload-meme
- download-meme
- react-meme
- delete-meme
- delete-user
- download-my-meme
- update-profile

Some key designs

- Custom DB Schema hosted on Cloud SQL server integrated with App Engine
- Memes are stored in the database in the form of MEDIUMBLOB which allows us to store memes upto size 16MB
- Memes are compressed to have fixed size for efficient storage in DB
- Memes (images) are encoded in base_64 and those data are stored in the DB. So, during, upload-meme and download meme, we have



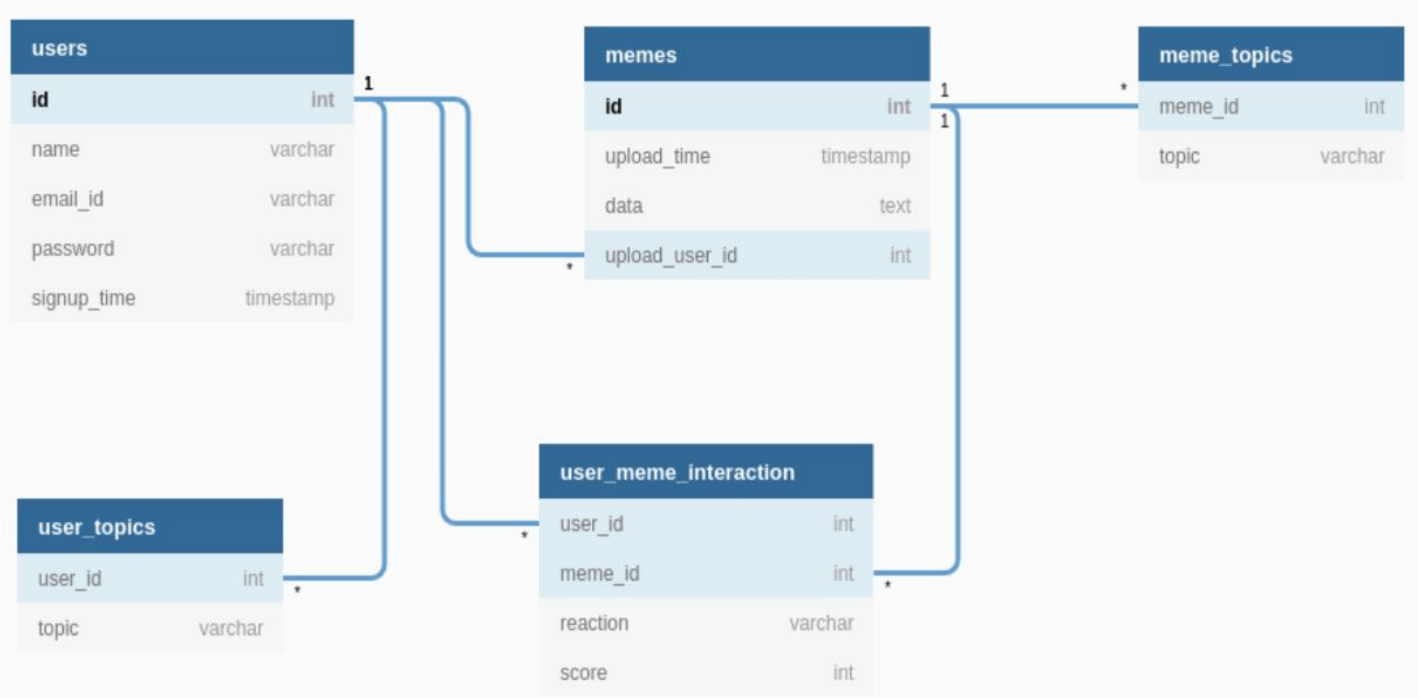
upload-meme

Image from local storage → Convert bytes to base_64 → Insert base_64 data into DB → Status OK

download-meme

Get data in the form of stream bytes in server-side → Convert bytes to chars (base_64) → Decode base_64 to image in client-side → Show image in android app

Database Schema



users	
id	int
name	varchar
email_id	varchar
password	varchar
signup_time	timestamp

memes	
id	int
upload_time	timestamp
data	text
upload_user_id	int

meme_topics	
meme_id	int
topic	varchar

user_topics	
user_id	int
topic	varchar

user_meme_interaction	
user_id	int
meme_id	int
reaction	varchar
score	int

Modified Hot Ranking Algorithm

Need

Jokes/memes need to be ranked based on user reactions, user topic interests, time of posting.

The Hot Ranking Algorithm used by Reddit to display trending posts has been modified according to the following for this application.

Initial formula

```
order(s) + sign(s) * seconds / 45000
where s=5*no_of_rofls+1*no_of_likes-1*no_of_dislikes,
      order(s)=log10(max(abs(s), 1))
      seconds = upload_date - start_date(when product was built)
```

** all weights would be modified according to performance

1. The first term calculates the score from the reactions on the joke/meme
2. The second term aims to give recent jokes/memes high scores

Modifications to incorporate user interests

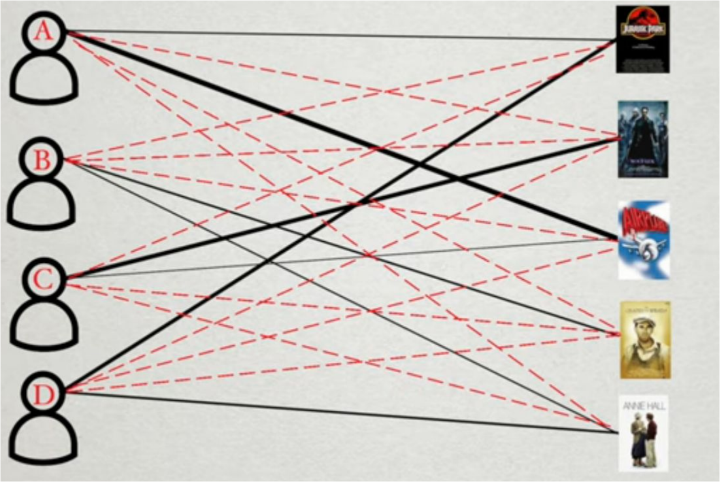
- Suppose we have a user u interested in a set of topics t_u
- Suppose we have a meme/joke m with topic tags set as t_m
- Let h be the score we get from hot ranking algo
- The number of topics tagged for meme/joke m: n=|t_m|
- For meme m, the number of topics user u is interested: n_i=|t_m∩t_u|
- For meme m, the number of topics user u is not interested: n_j=n-n_i

Final modified hot ranking algo score: h*(n_i+0.1*n_j)/n

Collaborative Filtering

WHAT?

- User preferences can be graphically represented as connections between people on one side and memes on the other
- Thick line => Higher preference
- Try and predict what the reaction (or a preference score) for a given unseen meme will be

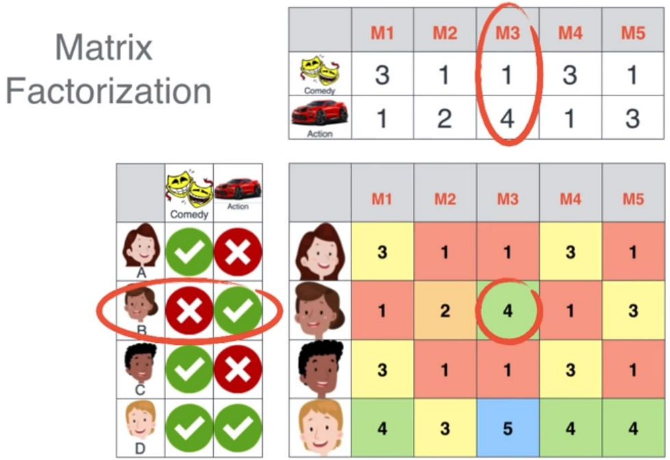


CONTENT FILTERING (OLD)

Given:

- Meme topics a user prefers at the time of signup
- Topics corresponding to particular

Weighted average gives the rating a user will give to a meme. Overly simplistic and doesn't consider hidden latent features



Matrix Factorization

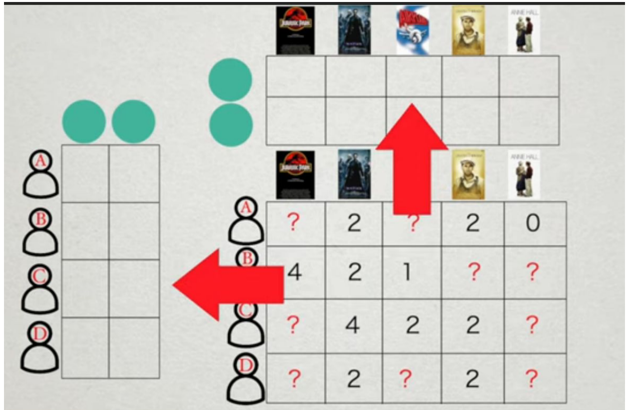
	M1	M2	M3	M4	M5
Comedy	3	1	1	3	1
Action	1	2	4	1	3

	Comedy	Action
A	✓	✗
B	✗	✓
C	✓	✗
D	✓	✓

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

HOW?

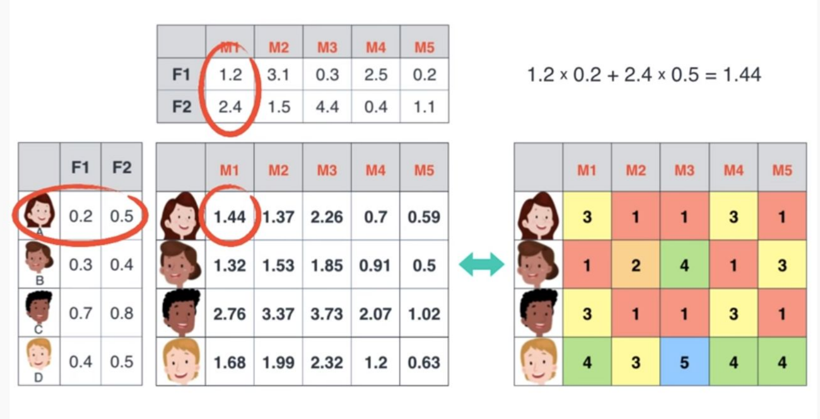
- Matrix with rows as users and columns as memes
- Value at given cell = user-meme interaction (like, dislike, rofl) = (1,-1,5)
- Generate features using the patterns in the incomplete set of preference data
- Approximate matrix factorization into 2 matrices
- Back propagation to get the exact factor matrices
- Use factor matrices to fill in the blank cells



A	?	2	?	2	0
B	4	2	1	?	?
C	?	4	2	2	?
D	?	2	?	2	?

OVERVIEW

- Motivation: Idea that you will probably like things that people with similar viewing habits also like.
- Here, we throw away the idea of dreaming up features, which were used to connect users and memes, in this case the meme topics.
- Used the user's past interactions with different memes to get a trained neural network model which predicts scores for each of the memes in the database.



	M1	M2	M3	M4	M5
F1	1.2	3.1	0.3	2.5	0.2
F2	2.4	1.5	4.4	0.4	1.1

1.2 × 0.2 + 2.4 × 0.5 = 1.44

	F1	F2
A	0.2	0.5
B	0.3	0.4
C	0.7	0.8
D	0.4	0.5

	M1	M2	M3	M4	M5
A	1.44	1.37	2.26	0.7	0.59
B	1.32	1.53	1.85	0.91	0.5
C	2.76	3.37	3.73	2.07	1.02
D	1.68	1.99	2.32	1.2	0.63

	M1	M2	M3	M4	M5
A	3	1	1	3	1
B	1	2	4	1	3
C	3	1	1	3	1
D	4	3	5	4	4

MODEL

- Constructed a collaborative neural network model using embedding on user meme interactions data
- Trained this model using MSE loss b/w actual & expected reaction as the loss function and ReLu activation function
- Optimized the model parameters, predicted scores for the given dataset and updated them in the sql database
- Note: The latent feature matrix being constructed need not represent the predefined meme topics

CODE:

```
train_dl = DataLoader(train_df[['user_id', 'meme_id', 'rating']])
model = CollabFNet(num_users, num_items, emb_size=100)
final_scores= train(model, epochs, train_dl, actual_ratings, lrs)
```

ADVANTAGES:

- Doesn’t need any domain knowledge as the embeddings are automatically learnt
- Not required to understand the content, which doesn’t necessarily tell the whole story.
- Captures the change in user interests over time directly
- Captures inherent subtle characteristics in case a user likes 2 unrelated memes.

LIMITATIONS:

- Model is trained for that particular instance of the database instead of incorporating just the new entries.
- Timestamp or the time at which the user reaction is recorded is not being considered

Learnings

1. Poorvi - Content, Collaborative Filtering, pytorch implementations

2. Riya - Ranking Algorithms, Tasks Scheduling

3. Satya - Node.js, App Engine (GAE deployment - main obstacle)

4. Shubham - Frontend (Using Flutter), API designs

General: Experience working in new platforms; integrating code in group projects

Future Work

- Hot Ranking: Build classifier to get meme topics instead of asking uploader to mention
- Collaborative Filtering:
    - Incorporate the time at which the meme is uploaded. Score should decrease with time.
    - Model training for just the new entries instead of factoring the entire matrix at a given instant.
- UI:
    - Make app cache friendly to increase speed and ease of navigation
    - Decode the image and store it in advance to increase its speed

Questions & Suggestions?

THANK YOU!

Software
Product Sprint