COLLABORATIVE RECOMMENDER SYSTEM WITH RESTful API

Similarity Calculation Used:

Cosine similarity: Measure of similarity between two documents (in our case two user) that measures the cosine of the angle in between them.

Why use cosine similarity:

- Performs very well when the data is in 0 to 1 range. I normalized the data to fall in zero to one range using min max scaler.
- Low time complexity, as an initial baseline and short deadline (6-8 hours), I chose to put more detail and thought in building and deploying the API with front end.
- Works well with sparse matrices.

What could be done next:

- Use a mix of supervised and unsupervised learning. i.e. cluster users into optimal clusters and then build models for such clusters.
- Experiment with dimensionality reduction methods like PCA and SVD.
- Based on time and trend select features that best represents the user's current interests.

Working at scale:

- Data Engineering:
    - Set up timely jobs to load new data into the user tables
    - Index entries by time to make calls faster.
- ML pipeline:
    - Enable parallel processing and continuous updates to the scores table.
    - Enable sparsity reduction techniques.
    - Investigate and use the features that best represent the user behavior using metrics like course completion ration, continuous view streak, test scores, user activity and more.
    - For example, a user matches another user, but that user is inactive discount the similarity score.
- User Facing ML:

- Optimize the suggestion display position on the screen by making use of usability metrics like eye tracking, intent to move vs. click conversion etc.

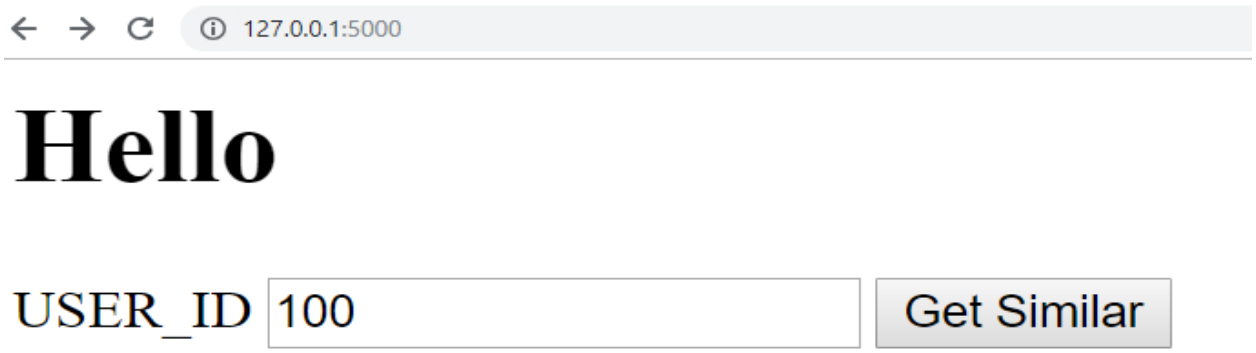
Other data that I would like to know:

- Number of active users' trend.
- Does test score mean anything?
- Incentive to complete/ stay logged on.
- Course author ratings are very important as most people turn to the best instructor available.
- Are there any course paths in the system for example a sequential list of courses towards a goal, this can be useful to group people together and increase the similarity ranking?

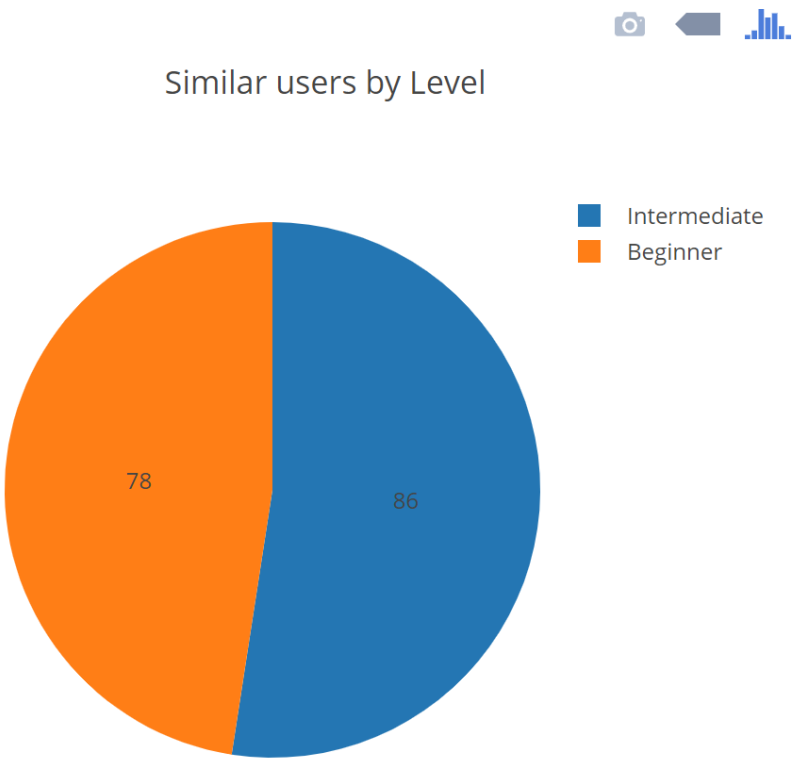
Application design steps:

- Explore the data set (1-2 hours): Played around with the data set to notice that it would be sparse as expected. Also, test assessment is of no use as they do not account for at least 50% of the users. Using the information would only increase sparsity.
- Design and create database functionality: I integrated SQL backend into the application as I think that it is the right way going forward, holding data frames in memory might prove costly later.
- The above step also includes calculation of similar users. In this version only 10 most similar users are retained.
- Set up application and testing basic functionality
- Set up a landing page to input the user query id and output dynamic Plotly graphs showing the summary of the most similar users.
- Adding write ups, making build and committing final commit.

Landing page:

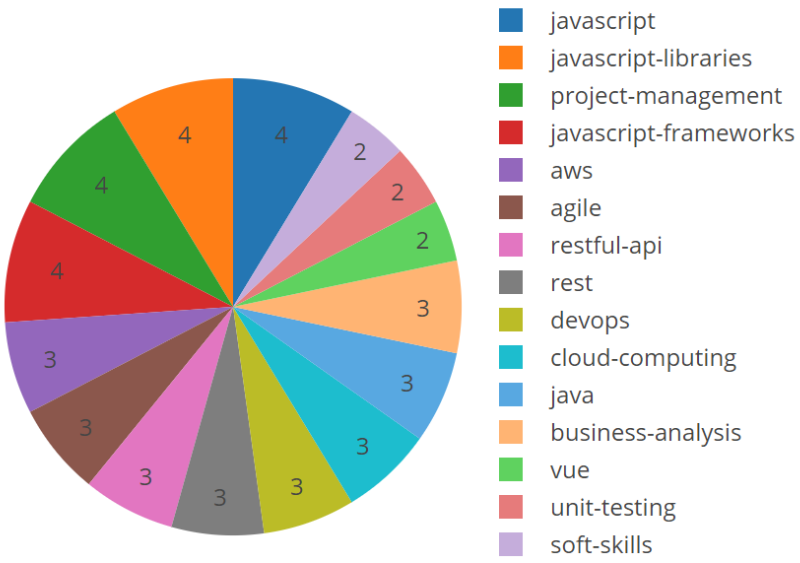


Dashboard page:

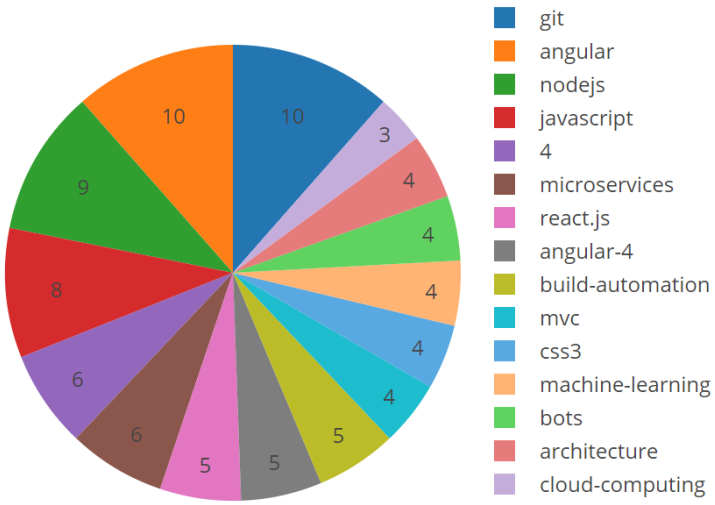


Edit chart »

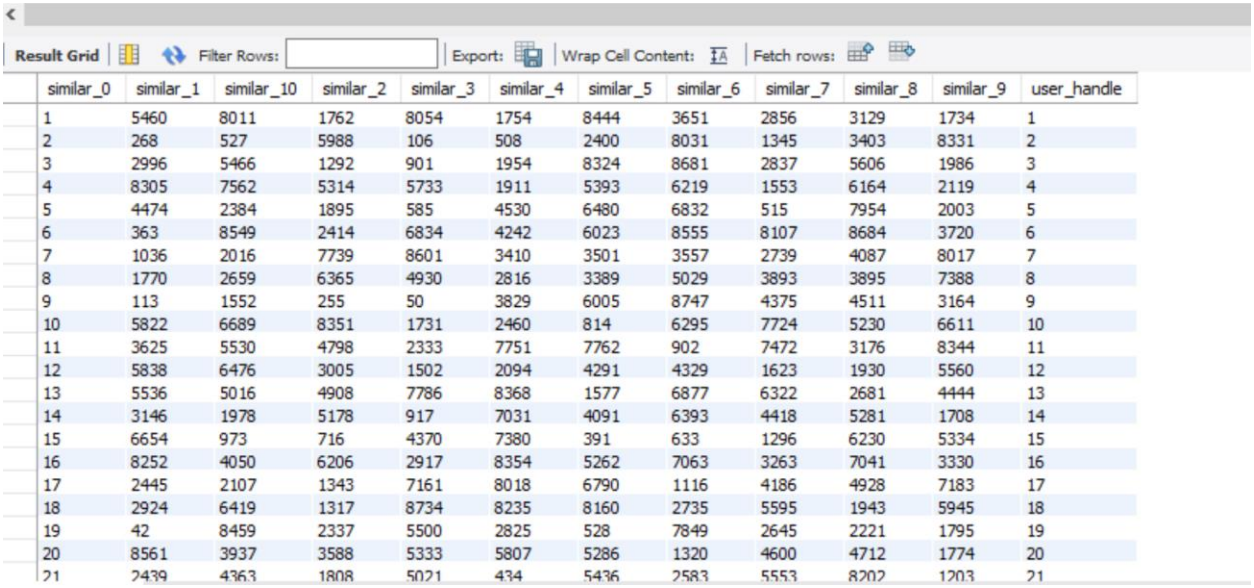
Similar users by Interests



Users similar to you viewed courses from the categories



Database similar users:



Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

similar_0	similar_1	similar_10	similar_2	similar_3	similar_4	similar_5	similar_6	similar_7	similar_8	similar_9	user_handle
1	5460	8011	1762	8054	1754	8444	3651	2856	3129	1734	1
2	268	527	5988	106	508	2400	8031	1345	3403	8331	2
3	2996	5466	1292	901	1954	8324	8681	2837	5606	1986	3
4	8305	7562	5314	5733	1911	5393	6219	1553	6164	2119	4
5	4474	2384	1895	585	4530	6480	6832	515	7954	2003	5
6	363	8549	2414	6834	4242	6023	8555	8107	8684	3720	6
7	1036	2016	7739	8601	3410	3501	3557	2739	4087	8017	7
8	1770	2659	6365	4930	2816	3389	5029	3893	3895	7388	8
9	113	1552	255	50	3829	6005	8747	4375	4511	3164	9
10	5822	6689	8351	1731	2460	814	6295	7724	5230	6611	10
11	3625	5530	4798	2333	7751	7762	902	7472	3176	8344	11
12	5838	6476	3005	1502	2094	4291	4329	1623	1930	5560	12
13	5536	5016	4908	7786	8368	1577	6877	6322	2681	4444	13
14	3146	1978	5178	917	7031	4091	6393	4418	5281	1708	14
15	6654	973	716	4370	7380	391	633	1296	6230	5334	15
16	8252	4050	6206	2917	8354	5262	7063	3263	7041	3330	16
17	2445	2107	1343	7161	8018	6790	1116	4186	4928	7183	17
18	2924	6419	1317	8734	8235	8160	2735	5595	1943	5945	18
19	42	8459	2337	5500	2825	528	7849	2645	2221	1795	19
20	8561	3937	3588	5333	5807	5286	1320	4600	4712	1774	20
21	2439	4363	1808	5021	434	5436	2583	5553	8202	1203	21

Improvements/ Next steps:

- Better similarity metrics
- Improved testing suite
- Acquire more data and set up a pipeline
- Improvise front end using dask