Recommender System Evaluation

Team Name: Bald Unicorn

Team Members: Manivannan Prushorth, Jun Yi Goh

Content

Instructions on how to reproduce results

0. **May need to run on MacOS – Due to insufficient RAM on Linux virtual machine**
1. **Go to this file:** bookClub/unibook/recommender/evaluation/evaluator.py in the main method – Suggest to evaluate one algorithm at a time (Warning – Might crash if attempt to evaluate multiple algorithms at same time) – See below for how to do this for our algorithms.
2. **On terminal –** make sure current directory is bookClub and virtual environment is activated.
3. **Terminal Command: python3 unibook/recommender/evaluation/evaluator.py**

What is to be Varied

1. ***K-Nearest-Neighbours (KNN) Algorithm:*** KNNBasic, KNNWithMeans, KNNBaseline KNNWithZScore

2. ***Similarity Measures:*** Cosine, Mean-Square-Difference (MSD), Pearson, Pearson-Baseline

User KNN Algorithm

1.Uncomment block labelled "BLOCK 1" in evaluator.py in main method
2.Vary userKNN = **KNNBasic**(sim_options={'name': **'pearson_baseline'**, 'user_based': True})
3. Run terminal command above
3. Comment block labelled "BLOCK 1"

Item KNN Algorithm

1. Same as User KNN Algorithm but uncomment "BLOCK 2"

Simple User Collaborative Algorithm

1. Uncomment block labelled "BLOCK 3"
2. Run terminal command above
3. Comment block labelled "BLOCK 3"

Simple Item Collaborative Algorithm

1. Same as Simple User Collaborative but uncomment "BLOCK 4"

SVD

1. Same as Simple User Collaborative but uncomment "BLOCK 5"

SVD++

1. Same as Simple User Collaborative but uncomment "BLOCK 6"

Content Year

1. Uncomment block labelled "BLOCK 7"
2. Go to recommender/evaluator/algorithms.py
3. Uncomment line 38 and 41
4. Run terminal command above
5. Comment line 38 and 41

Content Author

1. Same as Content – Year but with Lines 39 and 42

Content Publisher

1. Same as Content – Year but with Lines 40 and 43

Content Year and Author

1. Same as Content – Year but with Lines 38, 39, 44

Content Year and Publisher

1. Same as Content – Year but with Lines 38,40,45

Content Author and Publisher

1. Same as Content – Year but with Lines 39,40,46

Content Year and Author and Publisher

1. Same as Content – Year but with Lines 38,39,40,47

Spark

1. Make sure spark installed – See installation instructions report
2. Uncomment block labelled "BLOCK 8"
3. Run terminal command above - Results may vary to ours
4. Comment Block "8"

Random

1. Uncomment block labelled "BLOCK 9"
2. Run terminal command above
3. Comment "BLOCK 9"

Key for metrics in tables below

KNN: K-Nearest-Neighbours

RMSE: Root Mean Squared Error – Lower scores means better accuracy. E.g Generally of by 1.5* in a 1-10 rating

MAE: Mean Absolute Error – Lower score means better accuracy

HR: Hit Rate – Higher score is better. How often to recommend a left out rating

CHR: Cumulative Hit Rate – Higher score is better – ratings above a certain threshold

ARHR: Average Reciprocal Hit rate – Higher score is better – takes rankings into account

Coverage: Higher score is better.

Novelty: Higher scores means more novel.

Diversity: Higher means more diverse

Chosen Recommender System

SVD

Why

RMSE is the lowest out of all the results therefore this implies better accuracy and the predictions are generally off by about 1.55 stars using rating scale from (1-10) stars. The same argument can also be used for the MAE. We also looked at the top-10 recommendations produced and felt satisfied that it produced a good range of well-known books and books which may interest users having seen recommendations based on our interests. In terms of the hit rate we would want to be as high as possible with the highest recorded being recorded as 0.1317 when using Pearson-baseline. Looking at coverage it may not have been as high as algorithms such as User KNN but was still quite close. Diversity and novelty were also quite high compared to the rest of our results so we felt that this was the best algorithm to choose. When generating the recommendations, it was a lot quick compared to the others.

KNN-Basic – User KNN, Item KNN, Simple User Collab, Simple Item Collab

KNN BASIC	Sim Measure	RMSE	MAE	HR	CHR	ARHR	Coverage	Diversity	Novelty	Time (minutes)
User-KNN	Cosine	1.8869	1.4641	0.0010	0.0010	0.0002	1.0000	0.9371	1502.2002	10
	MSD	1.8501	1.4071	0.0010	0.0010	0.0002	1.0000	0.9767	1509.1527	10
	Pearson	1.9477	1.5010	0.0060	0.0060	0.0017	0.8026	1.0007	987.2982	10
	PearsonBaseline	1.9258	1.4629	0.0036	0.0036	0.0012	0.8617	1.0000	1010.5887	10
Item-KNN	Cosine	1.6891	1.2365	0.0031	0.0031	0.0009	0.8383	0.6528	1544.7292	10
	MSD	1.7047	1.2380	0.0039	0.0039	0.0013	0.8397	0.8379	1554.1521	10
	Pearson	1.7566	1.2819	0.0082	0.0082	0.0029	0.8426	0.9034	923.6385	10
	PearsonBaseline	1.7636	1.2767	0.0080	0.0080	0.0027	0.8429	0.9987	965.8362	10
Simple User	Cosine	NOT MEANT TO EVALUATE THIS		0.0669	NOT MEANT TO EVALUATE THIS		0.9394	0.9082	737.2661	10
	MSD			0.0719			0.9394	0.9635	722.1374	10
	Pearson			0.0669			0.9394	0.9987	770.1858	10
	PearsonBaseline			0.1096			0.9394	1.0000	760.1052	10
Simple Item	Cosine			0.0649			0.9394	0.1043	593.8292	10
	MSD			0.0415			0.9394	0.5366	908.9223	10
	Pearson			0.0867			0.9394	0.8168	627.7421	10
	PearsonBaseline			0.1371			0.9394	0.9961	483.6915	10

KNNWithMeans– User KNN, Item KNN, Simple User Collab, Simple Item Collab

KNN WITH MEANS	Sim Measure	RMSE	MAE	HR	CHR	ARHR	Coverage	Diversity	Novelty	Time-min
User-KNN	Cosine	1.6935	1.2623	0.0012	0.0012	0.0003	0.9517	0.9189	1522.1220	10
	MSD	1.7152	1.2731	0.0024	0.0024	0.0006	0.9607	0.9695	1483.6771	10
	Pearson	1.7587	1.3053	0.0060	0.0060	0.0017	0.8115	0.9993	1019.9143	10
	PearsonBaseline	1.7594	1.3032	0.0060	0.0060	0.0024	0.8506	1.0000	1045.8293	10 lostream message will still run
Item-KNN	Cosine	1.6525	1.2310	0.0046	0.0046	0.0010	0.9976	0.6212	1489.4558	10
	MSD	1.6708	1.2419	0.0046	0.0046	0.0009	0.9981	0.7606	1442.6234	10
	Pearson	1.7262	1.2888	0.0101	0.0101	0.0027	1.000	0.9272	1033.7538	10
	PearsonBaseline	1.7270	1.2846	0.0104	0.0104	0.0026	1.000	0.9972	1074.7924	10
Simple User	Cosine	NOT MEANT TO EVALUATE THIS		0.0669	NOT MEANT TO EVALUATE THIS		0.9394	0.9082	737.2661	5
	MSD			0.0719			0.9394	0.9635	722.1374	5
	Pearson			0.0669			0.9394	0.9987	770.1858	5
	PearsonBaseline			0.1096			0.9394	1.0000	760.1052	5
Simple Item	Cosine			0.0649			0.9394	0.1043	593.8292	5
	MSD			0.0415			0.9394	0.5366	908.9223	5
	Pearson			0.0867			0.9394	0.8168	627.7421	5
	PearsonBaseline			0.1371			0.9394	0.9961	483.6915	5

KNNBaseline– User KNN, Item KNN, Simple User Collab, Simple Item Collab

KNN BASELINE	Sim Measure	RMSE	MAE	HR	CHR	ARHR	Coverage	Diversity	Novelty	Time-min
User-KNN	Cosine	1.6992	1.2981	0.0017	0.0017	0.0004	0.9966	0.9170	1526.5757	10
	MSD	1.7110	1.2892	0.0027	0.0027	0.0006	0.9973	0.9693	1480.9736	10
	Pearson	1.7413	1.3288	0.0080	0.0080	0.0022	0.9500	1.0037	991.9290	10
	PearsonBaseline	1.7370	1.3091	0.0070	0.0070	0.0026	0.9628	1.0000	1017.4304	10
Item-KNN	Cosine	1.6100	1.1913	0.0048	0.0048	0.0011	0.9083	0.6050	1353.6647	10
	MSD	1.6300	1.1998	0.0053	0.0053	0.0014	0.9131	0.7573	1281.5365	10
	Pearson	1.6409	1.2119	0.0121	0.0121	0.0035	0.9203	0.8894	776.1524	10
	PearsonBaseline	1.6506	1.2127	0.0104	0.0104	0.0032	0.9225	0.9962	814.0950	10
Simple User	Cosine	NOT MEANT TO EVALUATE THIS		0.0669	NOT MEANT TO EVALUATE THIS		0.9394	0.9082	737.2661	5
	MSD			0.0719			0.9394	0.9635	722.1374	5
	Pearson			0.0669			0.9394	0.9987	770.1858	5
	PearsonBaseline			0.1096			0.9394	1.0000	760.1052	5
Simple Item	Cosine			0.0649			0.9394	0.1043	593.8292	5
	MSD			0.0415			0.9394	0.5366	908.9223	5
	Pearson			0.0867			0.9394	0.8168	627.7421	5
	PearsonBaseline			0.1371			0.9394	0.9961	483.6915	5

KNNWithZScore– User KNN, Item KNN, Simple User Collab, Simple Item Collab

KNN WITH Z SCORE	Sim Measure	RMSE	MAE	HR	CHR	ARHR	Coverage	Diversity	Novelty	Time-min
User-KNN	Cosine	1.6800	1.2386	0.0017	0.0017	0.0004	0.9194	0.9270	1536.8476	10
	MSD	1.7118	1.2556	0.0022	0.0022	0.0004	0.9259	0.9718	1480.3951	10
	Pearson	1.7421	1.2835	0.0080	0.0080	0.0024	0.7900	0.9994	1022.6932	10
	PearsonBaseline	1.7430	1.2813	0.0053	0.0053	0.0020	0.8265	1.0000	1057.8950	10
Item-KNN	Cosine	1.6636	1.2377	0.0039	0.0039	0.001	1.000	0.6437	1471.3667	10
	MSD	1.6857	1.2511	0.0039	0.0039	0.0009	1.000	0.7689	1425.5258	10
	Pearson	1.7366	1.2979	0.0082	0.0082	0.0021	1.0000	0.9309	1038.1263	10
	PearsonBaseline	1.7387	1.2947	0.0089	0.0089	0.0027	1.0000	0.9974	1069.3247	10
Simple User	Cosine	NOT MEANT TO EVALUATE THIS		0.0669	NOT MEANT TO EVALUATE THIS		0.9394	0.9082	737.2661	5
	MSD			0.0719			0.9394	0.9635	722.1374	5
	Pearson			0.0669			0.9394	0.9987	770.1858	5
	PearsonBaseline			0.1096			0.9394	1.0000	760.1052	5
Simple Item	Cosine			0.0649			0.9394	0.1043	593.8292	5
	MSD			0.0415			0.9394	0.5366	908.9223	5
	Pearson			0.0867			0.9394	0.8168	627.7421	5
	PearsonBaseline			0.1371			0.9394	0.9961	483.6915	5

SVD, Content, Spark, Random

	RMSE	MAE	HR	CHR	ARHR	Coverage	Diversity	Novelty	Time - min
SVD - done	1.5558	1.1869	0.0046	0.0046	0.0019	0.8139	0.9390	813.4727	4-5 minutes
SVD++ - done	1.5650	1.1884	0.0070	0.0070	0.0032	0.8496	0.9604	927.4054	10 Minutes lostream message but will still run
Content - Year	1.6184	1.2098	0.0017	0.0017	0.0006	0.2315	0.9750	1257.2040	10
Content Author	1.6664	1.2101	0.0478	0.0478	0.0195	0.7615	0.9625	948.1036	10
Content Publisher	1.7323	1.2633	0.0193	0.0193	0.0064	0.8001	0.9645	1044.6985	10
Content - Year and Author	1.6659	1.2092	0.0461	0.0461	0.0191	0.7601	0.9621	962.7594	10
Content Year and Publisher	1.7323	1.2631	0.0191	0.0191	0.0067	0.8006	0.9639	1053.1183	10
Content Author and Publisher	1.6852	1.2391	0.0427	0.0427	0.0190	0.6671	0.9631	952.5529	10
Content– Year, Author and Publisher	1.6850	1.2390	0.0418	0.0418	0.0183	0.6667	0.9627	960.3954	10
Spark	3.6808	2.6089	NOT MEANT TO EVALUATE THIS						Results may vary
Random – Normal Predictor	2.3641	1.8799	0.0029	0.0029	0.0007	1.0000	0.9704	1169.0887	10 minutes lostream message but will still run Results may vary

Section 2 Testing

2.0 Introduction:

We began writing unit tests on the 23rd of February, the unit tests were initially written in Django-Templates. We split our tests into three folders views, models, and forms. Our tests checked any redirects, templates used, status codes, and decorators (such as login prohibited). However, at the start of March, the team made the decision to convert our front-end to react and backend to Django-rest framework. That also means that we had to refactor all our unit tests to Django-rest framework. Although, it was a long process to refactor all the unit tests, the tests were quite easy to write and required less code. This is the case since Django rest does not use forms and templates therefore, we did not have to write tests for them. In Django rest, we wrote automated unit tests for all serializers, models, and views. However, for the front-end we don't have an automated unit tests as we mainly relied on manual testing.

2.1 Tests statistics:

Number of tests: 427

Passing: 427

Failing: 0

Errors: 0

2.2 Code coverage:

bookClub/urls.py	5	0	100%
frontend/admin.py	1	0	100%
frontend/apps.py	4	0	100%
frontend/models.py	1	0	100%
frontend/urls.py	3	0	100%
frontend/views.py	3	1	67%
unibook/admin.py	25	0	100%
unibook/apps.py	4	0	100%
unibook/models.py	76	7	91%
unibook/orm_helper/orm_book.py	46	21	54%
unibook/orm_helper/orm_club.py	61	44	28%
unibook/orm_helper/orm_user.py	43	19	56%
unibook/permissions.py	22	8	64%
unibook/recommender/algorithms.py	62	43	31%
unibook/recommender/book_data.py	118	30	75%
unibook/recommender/evaluation/dataset_evaluator.py	38	24	37%
unibook/recommender/evaluation/recommender_metrics.py	112	98	12%
unibook/recommender/recommender_system.py	25	5	80%
unibook/recommender/spark_recommender.py	59	42	29%
unibook/recommender/surprise_recommender.py	152	99	35%
unibook/serializers.py	120	31	74%
unibook/urls.py	8	0	100%
unibook/views.py	372	52	86%
TOTAL	1360	524	61%

2.3 Commands to get Code Coverage:

Firstly, install coverage by running pip3 install -r requirements.txt. Afterwards, run the command coverage run manage.py test, this will run all the tests. Finally, to get the coverage report run the command coverage report. This will return the coverage report as seen above.

- pip3 install -r requirements.txt
- coverage run manage.py test
- coverage report

2.4 Manual testing:

Our team mostly focused on automated testing however we still conducted manual testing for our user authorization. For manual testing please do not run the seed command as it will cause confusion over the club id.

User auth:

- Firstly, create a club by filling out the necessary information, this will make you the owner.
- Click on the club's info
- Go the admin interface and click on user auth.
- Change the user auth from owner to applicant in the club you created and save it.
- Afterwards, attempt to create a meeting in the club's info which should return HTTP 400 bad request.

Member cannot view applicant list:

- Firstly, apply to a club
- Go to the admin interface and login using a superuser you created.
- Go user_auth and change your user from applicant to member (basically accept yourself into the club).
- Type the URL applicant_list/club_id (usually 1)
- This should return HTTP 403 method not allowed

Front-end:

- As a logged in user, click on the top right profile icon. This should give you a drop-down menu including log out, change password, and edit profile.
- You can click on each one and it should redirect you to desired page.
- When on the dashboard, click on the side bar, this should extend and display rate books, my clubs, my meeting, create a club, and dashboard.
- Once books have been rated, the dashboard should display recommendations with the name and picture of the book the right-hand side of the dashboard. On the left-hand side, you will be able to see the clubs you can join.
- If no books are rated, the club list will be empty and so will the recommendations.

Section 3 Instructions:

- Firstly, create a virtual environment with the command **virtualenv venv**
- Activate it using the command **source venv/bin/activate**
- Install the required packages with the command **pip3 install -r requirements.txt**
- Make migrations with the command **python3 manage.py makemigrations**
- Apply them with the command **python3 manage.py migrate**
- You can seed the data base by **python3 manage.py seed**
- And run all the tests with the command **python3 manage.py test**
- For our home page video to load, you need to install Multimedia Codecs Ubuntu 20.04 LTS.
- Instructions can be found here https://linuxhint.com/install_multimedia_codecs_ubuntu/
- To use spark recommender system, Java 8 needs to be installed and used: Installation of java 8: https://docs.datastax.com/en/jdk-install/doc/jdk-install/installOpenJdkDeb.html
- Switching between java versions in ubuntu if multiple versions are installed: https://aboullaite.me/switching-between-java-versions-on-ubuntu-linux/
- To view recommendations, you need to rate some books in the rate books page which can be accessed from the side bar in the dashboard.