Recommender comparisons

Training data for all recommenders are comprised of all Compass data until 4/28/2018 for learners with 10 or more video's watched.

Predictions are generated for all users and videos based on their history of 10 or more video's during that period.

Test set is comprised of all compass data from 4/28/2018 until 7/13/2018 - excluding learners without observations before 4/28

TEST criteria:

- RMSE based on the difference between the predicted like rating and the actual like rating
- Proportion of videos in the top 50 recommended videos that were watched by learners
- Proportion of learners that watched any of the 50 highest recommended videos

The proportion watched criteria are biased, as the current recommender heavily influences which videos learners watch, so we expect our existing recommender to do much better there.

In addition, the rmse might also be influence by the fact that the current recommendations are heavily restricted by word cards etc.

Recommender	Proportion of videos viewed in top 50	Proportion of learners with at least 1 watched video in top 50 recommender	Mean rank of viewed items	Median of mean rank of viewed videos per learner	RMSE	Number of learners without predictions	Total number of learners	25th percentile number of videos watched in test set	Median number of videos watched in test set	75th percentile number of videos watched in test set
current	0.0443 (e.g. 2.2 videos out of 50 recommended)	0.1516	7949.4	8534.0	0.6752	142	4166	2	3	7

hybrid_GB	0.0021	0.0097	4251.6	4367.3	**0.6291**	0	4041	2	3	8
hybrid_EN	0.0017	0.0097	4127.5	4256.8	0.6300	0	4041	2	3	8
KNN_user	0.0000	0.0000	4093.6	4122.3	0.6378	0	4041	2	3	8
KNN_item	0.0000	0.0000	4156.8	4240.9	0.6437	0	4041	2	3	8
Baseline_ Only	0.0000	0.0000	4161.2	4240.9	0.6385	0	4041	2	3	8
SVD	0.0032	0.0193	4027.9	4057.0	0.6877	0	4041	2	3	8
SVD++	0.0056	0.0250	3986	4042						

PROBLEM

Ok so here is a problem (already indicated by the relatively good performance of the baseline_only recommender): just recommending the most popular videos will get a great RMSE, even though the same set of videos is recommended to everyone.

Number of unique learners

```
> length(unique(predictions[[1]]$learner_id))
[1] 5784
```

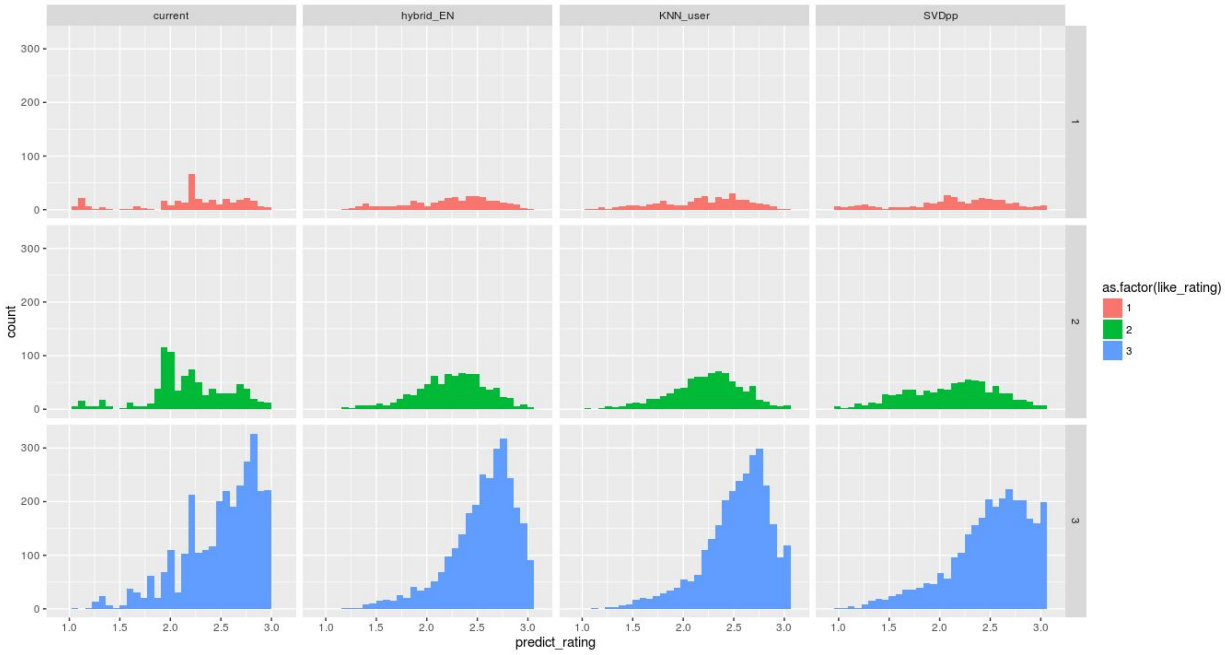
Number of unique videos:

```
> length(unique(predictions[[1]]$media_id))
[1] 13654
```

Recommender	Number of distinct media_ids in top50 recommended items:	
current	3038	
hybrid_GB	425	
hybrid_EN	1688	
KNN_user	76	
KNN_item	95	
Baseline_Only	**55**	ALL USERS BASICALLY GET THE SAME ITEMS !!!!

SVD	367	
SVD++	587	

So… maybe the RMSE is not our only/best metric of choice
Maybe we should prioritize predicting the



Description of each recommender

Baseline_Only

Typical CF data exhibit large user and item effects—systematic tendencies for some users to give higher ratings than others—and for some items to receive higher ratings than others Baseline_Only is an algorithm to predict the baseline estimate for given user and item.

$$b_{ui} = \mu + b_u + b_i.$$

The parameters bu and bi indicate the observed deviations of user u and item i, respectively, from the average.

If user/item is unknown, then the respective bias is assumed to be zero.
To estimate $\mathbf{b_u}$ and $\mathbf{b_i}$ the least squares problem is solved using ALS or SGD

$$\min_{b_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2 \right)$$

KNN Baseline User & Item Based

User based methods estimate unknown ratings based on recorded ratings of like-minded users. Item based methods, a rating is estimated using known ratings made by the same user on similar items.
Similarity measure often based on Pearson correlation coefficient, $\boldsymbol{\rho_{ij}}$, which measures the tendency of users to rate items i and j similarly.
Similarities based on a greater user support are more reliable. An appropriate similarity measure, denoted by $\mathbf{s_{ij}}$, would be a shrunk correlation coefficient:

$$s_{ij} \stackrel{\text{def}}{=} \frac{n_{ij}}{n_{ij} + \lambda_4} \rho_{ij}.$$

The variable $\mathbf{n_{ij}}$ denotes the number of users who rated both i and j

Using the similarity measure, we identify the k items rated by u, which are most similar to i. This set of k neighbors is denoted by $\mathbf{S^k(i; u)}$
The predicted value of $\mathbf{r_{ui}}$ is taken as a weighted average of the ratings of neighboring items, while adjusting for user and item effects through the baseline estimates

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j\in \mathrm{S}^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j\in \mathrm{S}^k(i;u)} s_{ij}}.$$

Latent factor models(SVD, SVD++)

Latent factor models comprise an alternative approach to collaborative filtering with the more holistic goal to uncover latent features that explain observed ratings.
A typical model associates each user u with a user-factors vector $\mathbf{p_u \in R^f}$, and each item i with an item-factors vector $\mathbf{q_i \in R^f}$.
In **SVD**, the prediction is done by taking an inner product:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

To estimate all the unknown, the following regularized squared error is minimized:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2 \right)$$

The minimization is performed by a very straightforward stochastic gradient descent:

$$\begin{aligned} b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \end{aligned}$$

where $e_{ui} = r_{ui} - \hat{r}_{ui}$

These steps are performed over all the ratings of the train set and repeated n_epochs times. Baselines are initialized to 0. User and item factors are randomly initialized according to a normal distribution

The **SVD++** algorithm, an extension of SVD taking into account implicit ratings. The prediction rating is set as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

Where
$\mathbf{I_u}$: the set of all items rated by user u
$\mathbf{y_j}$ terms are a new set of item factors that capture implicit ratings.
An implicit rating describes the fact that a user u rated an item j, regardless of the rating value.

Hybrid Recommenders

Baseline_Only, Knn_UserBased_Baseline, Knn_ItemBased_Baseline, SVD_biased and SVDpp_biased algorithms were used to generate recommenders for last 5 items held out for 9813 users.
These generated recommendations were used as features of the following hybrid recommenders:

- Elastic Net Linear Regression

- Gradient Boosting Regressor

Execution Report

python generate_kfold_recommendations.py configs/selected_recommenders.pickle --testing

Command being timed: "python generate_kfold_recommendations.py configs/selected_recommenders.pickle --testing"
User time (seconds): 733.35
System time (seconds): 3.80
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 12:17.28
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 4909168
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 88
Minor (reclaiming a frame) page faults: 333813
Voluntary context switches: 170
Involuntary context switches: 692
Swaps: 0
File system inputs: 54528
File system outputs: 125432
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0

algo_name	RMSE	no_of_all_predictions	time_taken	
BaselineOnly_SGD_Tuned	0.6939	49065	11.1570	sec
Knn_UserBased_Baseline_SGD_Tuned	0.6905	49065	2.4335	min
Knn_ItemBased_Baseline_SGD_Tuned	0.694	49065	31.3508	sec
SVD_biased_Tuned	0.7075	49065	24.1189	sec
SVDpp_biased_Tuned	0.7066	49065	8.6339	min

##

python hybrid_recommender_generate_data.py configs/selected_recommenders.pickle model_testing/

Command being timed: "python hybrid_recommender_generate_data.py configs/selected_recommenders.pickle model_testing/"

User time (seconds): 1.33
System time (seconds): 0.06
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:01.41
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 102464
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 41690
Voluntary context switches: 19
Involuntary context switches: 9
Swaps: 0
File system inputs: 8
File system outputs: 20752
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0

##

python generate_recommendations.py configs/selected_recommenders.pickle --train data/latest_rating.csv --predict

trainset n_users : 9813, n_users : 7829, n_ratings : 260767
Trained BaselineOnly_SGD_Tuned. Time Taken : 0.3820 sec
Trained Knn_UserBased_Baseline_SGD_Tuned. Time Taken : 12.9011 sec
Trained Knn_ItemBased_Baseline_SGD_Tuned. Time Taken : 2.5353 sec
Trained SVD_biased_Tuned. Time Taken : 1.6812 sec
Trained SVDpp_biased_Tuned. Time Taken : 50.0974 sec
Trained All Recommenders. Time Taken : 1.1266 min

Preparing Anti TestSet...
testset n_users : 9813, n_users : 7829, n_ratings : 76565210
Built Anti Testset. Time Taken : 2.6866 min
Generated Anti TestSet. Time Taken : 4.1002 min

Generated Anti TestSet Predictions. Time Taken : 3.3288 hr

Captured Recommender Predictions and Top N recommendations. Time Taken : 12.3763 min

Command being timed: "python generate_recommendations.py configs/selected_recommenders.pickle --train data/latest_rating.csv --predict"

User time (seconds): 13002.08
System time (seconds): 36.01
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 3:37:26
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 30842588
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 144312
Minor (reclaiming a frame) page faults: 13075536
Voluntary context switches: 10020
Involuntary context switches: 13854
Swaps: 0
File system inputs: 3218024
File system outputs: 27354304
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0

##
python hybrid_recommender_generate_recommendations.py configs/selected_recommenders.pickle

Loading Train Data hybrid_recommender/all_combined_predictions.csv… (49065, 5)
Loading Test Data results/anti_test_set_predictions.csv… (76565210, 5)
Trained and Predicted ElasticNet. Time Taken : 0.7268 sec
Captured Recommender Predictions and Top N recommendations. Time Taken : 2.0431 min

Trained and Predicted GradientBoostingRegressor. Time Taken : 24.9651 sec
Captured Recommender Predictions and Top N recommendations. Time Taken : 2.1665 min

Command being timed: "python hybrid_recommender_generate_recommendations.py configs/selected_recommenders.pickle"

User time (seconds): 275.47
System time (seconds): 41.61
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 5:17.82
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 25020268
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 2690
Minor (reclaiming a frame) page faults: 8557828
Voluntary context switches: 4944
Involuntary context switches: 1389
Swaps: 0
File system inputs: 1907720
File system outputs: 5886584

References

DETAILS FOR EVALUATION:

##

I included all events < 2018-04-28 UTC (~90% of the events), and will test predicted like ratings against actual like ratings >= 2018-04-28.

Predicted ratings from our recommender are at:

/home/webb/Compass_Recommender/compass_existinglearner_recommender_predict_ratings.csv

See: /home/webb/Compass_Recommender/compass_existinglearner_recommender.R lines 345+

```
...
test_events = readRDS('test_events.rds')
nrow(test_events)
9077

recs_vs_test_events = merge(test_events,recs,by=c("learner_id","media_id"),all=F)
nrow(recs_vs_test_events)
4308 # because learners w < 10 like_ratings in events.rds excluded

nrow(recs_vs_test_events[!is.na(predict_rating)])
4166 # because videos w no like_ratings / plannels w no estimated like_ratings (?)

recs_vs_test_events[!is.na(predict_rating),sqrt(mean((like_rating-predict_rating)^2))]
0.6751592

write.csv(recs[,.(learner_id,media_id,predict_rating)],"compass_existinglearner_recommender_predict_ratings.csv",quote=F,row.names=F)
```

##########

Josine,

In order to compare with existing compass recommender, please use the following csv files which are recommendation scores generated for
anti_test_set(User, Item pairs not present in given data) *_predictions.csv and top 50 recommendations for each user in *_top_n_recs.csv

/home/ravi/analytics-tools/Compass_Recommender/ratings_recommender/top_n_recs/Hybrid_GradientBoostingRegressor_predictions.csv
/home/ravi/analytics-tools/Compass_Recommender/ratings_recommender/top_n_recs/Hybrid_GradientBoostingRegressor_top_n_recs.csv

/home/ravi/analytics-tools/Compass_Recommender/ratings_recommender/top_n_recs/Hybrid_ElasticNet_predictions.csv
/home/ravi/analytics-tools/Compass_Recommender/ratings_recommender/top_n_recs/Hybrid_ElasticNet_top_n_recs.csv

/home/ravi/analytics-tools/Compass_Recommender/ratings_recommender/top_n_recs/Knn_UserBased_Baseline_SGD_Tuned_predictions.csv
/home/ravi/analytics-tools/Compass_Recommender/ratings_recommender/top_n_recs/Knn_ItemBased_Baseline_SGD_Tuned_predictions.csv
/home/ravi/analytics-tools/Compass_Recommender/ratings_recommender/top_n_recs/BaselineOnly_SGD_Tuned_predictions.csv
/home/ravi/analytics-tools/Compass_Recommender/ratings_recommender/top_n_recs/SVD_biased_Tuned_predictions.csv
/home/ravi/analytics-tools/Compass_Recommender/ratings_recommender/top_n_recs/SVDpp_biased_Tuned_predictions.csv

Following is R script was used to import rds events data to csv:

```
library(data.table);
datadir <- '/home/webb/Compass_Recommender/'
print("Loading Events Data...")
content  = readRDS(paste0(datadir,'content.rds'))
events   = readRDS(paste0(datadir,'events.rds'))
learners = readRDS(paste0(datadir,'learners.rds'))

events = merge(events, content,  by='media_id',  all=F)
events = merge(events, learners, by='learner_id',all=F)

dir.create('data')
csv_file_name <- 'data/events.csv'

print(paste("Writing to : ", csv_file_name) )
write.csv(events, file='data/events.csv')
```

Let me know if you face any issues

##############################

The selection of A, B, or C is based on <ALP learner.id> mod 3 equaling 0, 1, or 2, respectively.

ABC test evaluation: how many items do they watch? How many of the recommended items do they pick?