INFSCI 2480 Adaptive Information Systems
Final Project Report
Movie Recommendation System

Raphael Fernandes   Nishchal Nigam   Swapnil Asawa

System Introduction:

Our final project for the IS2480 course is a movies and TV shows web-based adaptive real-time recommendation system. It is a website hosted locally which asks the user to select a minimum of 10 movies, and rate them on a scale of 1-5. The system is based on collaborative filtering, and uses singular value decomposition (SVD). SVD worked best for the application of various algorithms we tested.

System User Interface:

We started the user interface with a basic php and mySQL to understand the navigations. We then decided to move to MEAN stack development. MEAN stands for MongoDB, ExpressJS, Angular, and NodeJS, which results in a sophisticated user interface, aesthetic, user friendly. Our UI consists of a table which consists of a list of movies with year, genre and the option to rate each available movie. The user has also the possibility to filter the movies table, hence searching for movies are easy. After selecting and rating a movie the system will append to a secondary table that holds the user selected movies with the possibility to remove any added movie in case it was mistakenly added. When the user clicks on the button "Get Recommendations!" the front-end will send an HTTP request to the server and will wait for the recommendation list as a return.

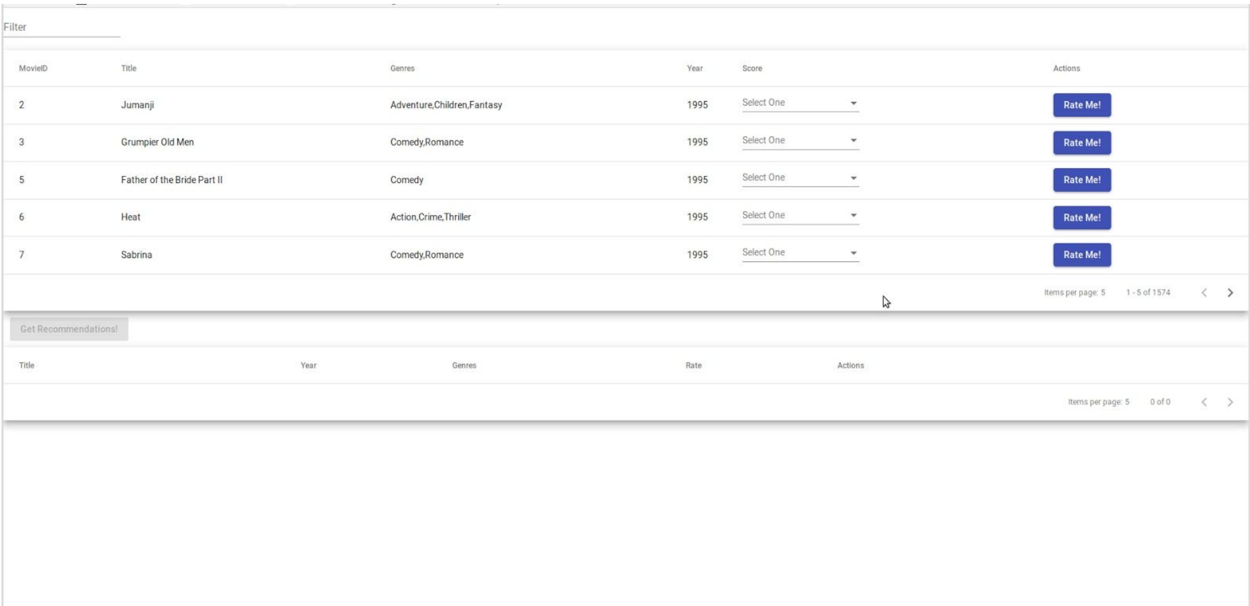


Figure 1: Home Page

Figure 2: Choosing Movies

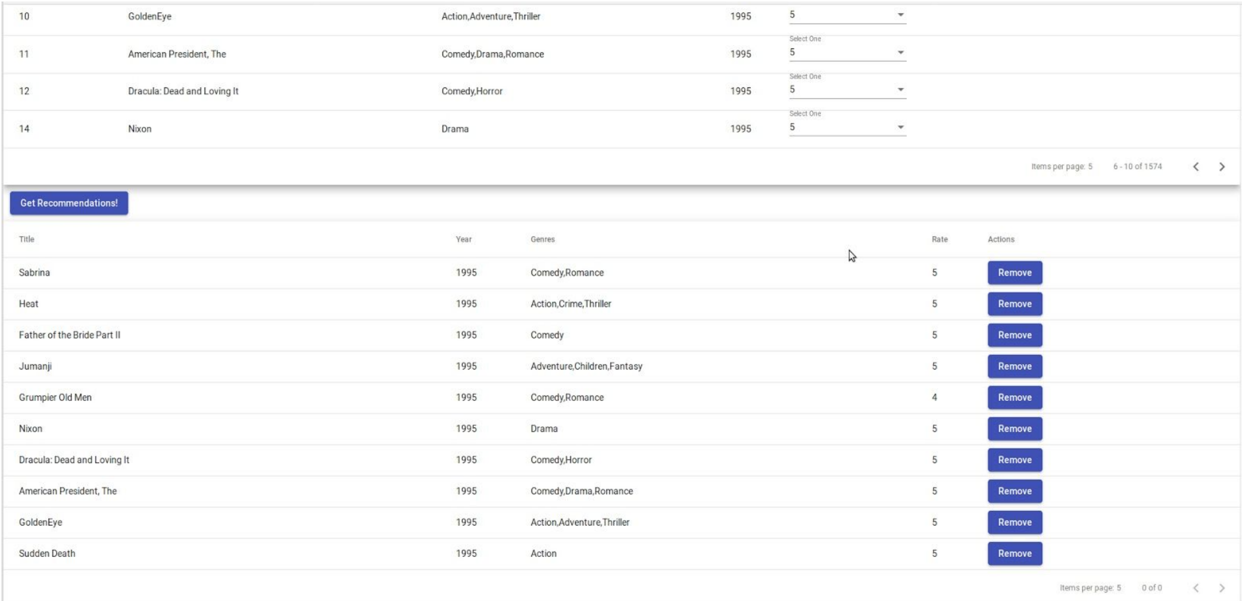


Figure 3: List with selected movies

Idea:

Our goal is to give personalised movie recommendation to a user who has rated few movies. The idea was evolved from our initial idea of “recommending travel destinations from movies preferences”, however, it would have used very primitive techniques of filtering and recommendation, giving less accurate results, and showcasing less about what we learned in the class. We thus decided to recommend movies from user movie preference, which uses many of the techniques of what we have learnt in this course. We use the IMDB database and make a movie recommendation system using SVD a Non-Probabilistic Collaborative Filtering Algorithm. As suggested in class, Content based techniques are suitable for documents like news articles and research papers, but does not give good results for music, videos or movies.

We finally evolved with the idea of recommending movies using movies rating dataset which used a lot of research we did in our previous idea. We used movielens dataset with 100k ratings, as well as movielens 27 million ratings to train and test our results. In this website people can view movies names, check for ones they like, give them ratings. Once the user has rated certain 10 movies, he/she can ask for recommendations. The system fits the model to the new user ratings and able to produce the results in few seconds in movielens 100k dataset and 6-10 minutes for large dataset. This is reasonable due to the amount of ratings and the computer that was being used to run the code.

The dataset:

The 100k dataset has 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users which was last updated on the grouplens website on 9/2018.

The 27 million dataset has 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users. Includes tag genome data with 14 million relevance scores across 1,100 tags. Last updated 9/2018.
We used 100k dataset for evaluating different algorithms and large dataset on the best algorithm found.

We removed extremely high and low rated movies with filtering. Movies that had the number of ratings less than 4,000 and more than 50,000 was removed from the computation. We also filtered out users with less than 200 ratings in the dataset. These steps resulted in 1,574 movies, and 10,363,450 ratings.

Algorithms:

To test and compare different algorithms, we tried on movielens 100k dataset.

KNN with means:

KNN with means subtract the ratings from the mean of the ratings for K nearest users before calculating ratings of user u.

KNN with means uses the following formula for calculating ratings for a user u, movie i. This following formula is on basis of user-user similarity:

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v) \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}$$

KNN with means formula on the basis of item-item similarity:

$$\hat{r}_{ui} = \mu_i + \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j) \cdot (r_{uj} - \mu_j)}{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j)}$$

KNN with z-scores:

KNN with z scores is similar to KNN with means, except it also divide the ratings by the standard deviations to find the z score. Uses the following formula for calculating ratings for a user u, movie i:

User-user similarity:

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v) \cdot (r_{vi} - \mu_v)/\sigma_v}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}$$

Item-item similarity:

$$\hat{r}_{ui} = \mu_i + \sigma_i \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j) \cdot (r_{uj} - \mu_j)/\sigma_j}{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j)}$$

Singular-value Decomposition

The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler.

The prediction $\hat{r}_{ui}$ is set as:

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

The equation we are trying to solve:

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}}$$

The algorithm tries to minimize the following, taking care of overfitting by regularization:

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

We kept the learning rates to be **0.005** and regularization terms are set to **0.02 and epoches to 20.**

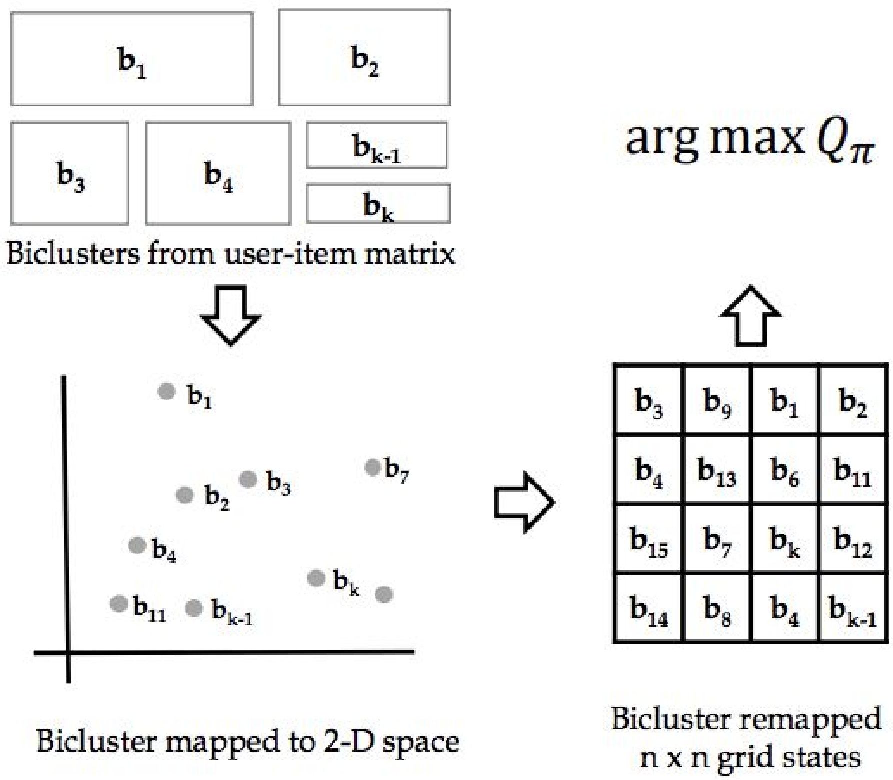
The minimization is performed by a very straightforward **stochastic gradient descent:**

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

where eui=rui−rˆui.
These steps are performed over all the ratings of the train set and repeated **n_epochs** times.

Reinforcement Learning based Recommender System using Biclustering Technique:

User Items matrix is first bi-clustered, and then mapped to a 2D euclidean space based on the difference of user vectors.



The reward of an action is calculated with Jaccard Distance.

$$R(s_t, a_t, s_{t+1}) = \text{Jaccard\_Distance}(U_{s_t}, U_{s_{t+1}})$$
$$= \frac{|U_{s_t} \cap U_{s_{t+1}}|}{|U_{s_t} \cup U_{s_{t+1}}|}.$$

The implementation of the method is incomplete in the limited time we had, thus we experiment our tests on the 1st three methods.

Experiments:

We first experimented on movielens-100k dataset to compare algorithms.

KNN user based on movielens-100k

Evaluating RMSE, MAE of algorithm KNNWithMeans on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.8916	0.8931	0.9032	0.8921	0.8919	0.8944	0.0045
MAE (testset)	0.6787	0.6787	0.6836	0.6763	0.6765	0.6788	0.0026
Fit time	0.57	0.59	0.59	0.59	0.46	0.56	0.05
Test time	1.39	1.37	1.37	1.38	1.31	1.36	0.03

KNN item based on movielens-100k

Evaluating RMSE, MAE of algorithm KNNWithMeans on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.8746	0.8791	0.8834	0.8772	0.8877	0.8804	0.0046
MAE (testset)	0.6635	0.6685	0.6721	0.6681	0.6723	0.6689	0.0032
Fit time	11.18	10.21	9.96	9.43	9.06	9.97	0.73
Test time	7.06	6.56	6.35	6.73	6.55	6.65	0.24

KNN with Z scores User Based:

Evaluating RMSE, MAE of algorithm KNNWithZScore on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.8870	0.8875	0.8835	0.8930	0.8838	0.8870	0.0034
MAE (testset)	0.6742	0.6741	0.6722	0.6765	0.6733	0.6740	0.0014
Fit time	10.69	10.86	10.00	10.11	9.82	10.30	0.40
Test time	11.87	11.46	11.46	12.07	10.95	11.56	0.39

KNN with Z scores Item-Item based.

Evaluating RMSE, MAE of algorithm KNNWithZScore on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9049	0.8900	0.8833	0.8922	0.8909	0.8923	0.0070
MAE (testset)	0.6791	0.6681	0.6653	0.6746	0.6683	0.6711	0.0050
Fit time	0.60	0.61	0.53	0.61	0.63	0.59	0.03
Test time	1.38	1.37	1.40	1.42	1.40	1.40	0.02

SVD on movielens-100k

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8652  0.8798  0.8744  0.8720  0.8734  0.8730  0.0047
MAE (testset)     0.6664  0.6751  0.6719  0.6722  0.6717  0.6715  0.0028
Fit time          4.81    4.81    4.81    4.67    4.65    4.75    0.07
Test time         0.18    0.16    0.17    0.16    0.16    0.17    0.01
```

We see SVD works better than all other algorithms. So we finally choose SVD and use that in the large datasets which takes more time in fitting and evaluation.
We use SVD with new user who input ratings online in our website.

Tests and results:

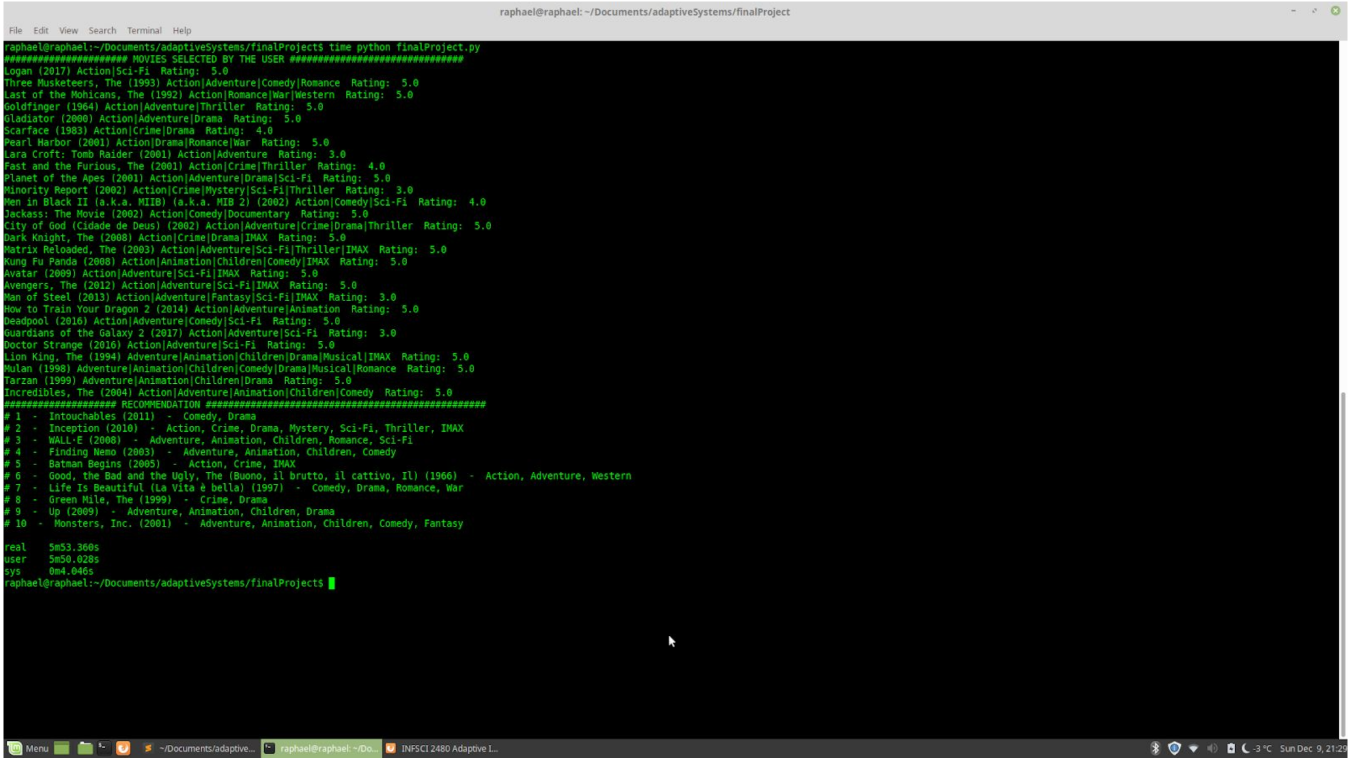
Our first test was using a mix of movies that consists of the following list and rating by movie:

Logan (2017) Action|Sci-Fi Rating: 5.0
Three Musketeers, The (1993) Action|Adventure|Comedy|Romance Rating: 5.0
Last of the Mohicans, The (1992) Action|Romance|War|Western Rating: 5.0
Goldfinger (1964) Action|Adventure|Thriller Rating: 5.0
Gladiator (2000) Action|Adventure|Drama Rating: 5.0
Scarface (1983) Action|Crime|Drama Rating: 4.0
Pearl Harbor (2001) Action|Drama|Romance|War Rating: 5.0
Lara Croft: Tomb Raider (2001) Action|Adventure Rating: 3.0
Fast and the Furious, The (2001) Action|Crime|Thriller Rating: 4.0
Planet of the Apes (2001) Action|Adventure|Drama|Sci-Fi Rating: 5.0
Minority Report (2002) Action|Crime|Mystery|Sci-Fi|Thriller Rating: 3.0
Men in Black II (a.k.a. MIIB) (a.k.a. MIB 2) (2002) Action|Comedy|Sci-Fi Rating: 4.0
Jackass: The Movie (2002) Action|Comedy|Documentary Rating: 5.0
City of God (Cidade de Deus) (2002) Action|Adventure|Crime|Drama|Thriller Rating: 5.0
Dark Knight, The (2008) Action|Crime|Drama|IMAX Rating: 5.0
Matrix Reloaded, The (2003) Action|Adventure|Sci-Fi|Thriller|IMAX Rating: 5.0
Kung Fu Panda (2008) Action|Animation|Children|Comedy|IMAX Rating: 5.0
Avatar (2009) Action|Adventure|Sci-Fi|IMAX Rating: 5.0
Avengers, The (2012) Action|Adventure|Sci-Fi|IMAX Rating: 5.0
Man of Steel (2013) Action|Adventure|Fantasy|Sci-Fi|IMAX Rating: 3.0
How to Train Your Dragon 2 (2014) Action|Adventure|Animation Rating: 5.0

Deadpool (2016) Action|Adventure|Comedy|Sci-Fi Rating: 5.0
Guardians of the Galaxy 2 (2017) Action|Adventure|Sci-Fi Rating: 3.0
Doctor Strange (2016) Action|Adventure|Sci-Fi Rating: 5.0
Lion King, The (1994) Adventure|Animation|Children|Drama|Musical|IMAX Rating: 5.0
Mulan (1998) Adventure|Animation|Children|Comedy|Drama|Musical|Romance Rating: 5.0
Tarzan (1999) Adventure|Animation|Children|Drama Rating: 5.0
Incredibles, The (2004) Action|Adventure|Animation|Children|Comedy Rating: 5.0

The recommendations were:

1 - Intouchables (2011) - Comedy, Drama
2 - Inception (2010) - Action, Crime, Drama, Mystery, Sci-Fi, Thriller, IMAX
3 - WALL·E (2008) - Adventure, Animation, Children, Romance, Sci-Fi
4 - Finding Nemo (2003) - Adventure, Animation, Children, Comedy
5 - Batman Begins (2005) - Action, Crime, IMAX
6 - Good, the Bad and the Ugly, The (Buono, il brutto, il cattivo, Il) (1966) - Action, Adventure, Western
7 - Life Is Beautiful (La Vita è bella) (1997) - Comedy, Drama, Romance, War
8 - Green Mile, The (1999) - Crime, Drama
9 - Up (2009) - Adventure, Animation, Children, Drama
10 - Monsters, Inc. (2001) - Adventure, Animation, Children, Comedy, Fantasy



Future Works:

Reinforcement Learning based Recommender System using Biclustering Techniques:

We tried to implement Reinforcement Learning using Biclusters of the user item matrix

Division of work:

Nishchal: Mainly idea of travel recommendation and Collaborative Filtering technique on movie recommendation. Research on techniques to be used, documentation, and assist on back-end communication. Testing to check if recommendations were consistent.

Raphael: Data collection, recommender algorithm development, and testing on different techniques.

Swapnil: Gave insights showing that our initial idea would not generate good recommendations and it was not clear enough for the final project, literature analysis, assist on front-end coding, documentation, and testing on different techniques.

References:

Dataset:
https://grouplens.org/datasets/movielens/

SVD:
http://buzzard.ups.edu/courses/2014spring/420projects/math420-UPS-spring-2014-gower-netflix-SVD.pdf

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.6458&rep=rep1&type=pdf

http://www.albertauyeung.com/post/python-matrix-factorization/

Python Library:
https://surprise.readthedocs.io/en/stable/index.html

Recommendations with Reinforcement Learning
https://arxiv.org/pdf/1801.05532.pdf