Milestone-2

Research Question: Which is a better Collaborative Filtering approach for building a recommendation engine for an Ecommerce platform like Amazon?

Given that the Recommendation engine can be designed using a lot of approaches, we have decided to work on 4 different models under the Collaborative Filtering approach.

1. K - Nearest Neighbors
2. Deep Learning based Neural Collaborative Filtering
3. Stochastic Gradient Descent
4. Alternating Least Squares

Work Distribution:

KNN and clustering approach is handled by **Swapnil**. Matrix factorization involving SGD and ALS are handled by **Sumanth** and **Vijay**. Deep Learning based approach involving NCF is handled by **Ajinkya**.

Data collection:

We are using the Amazon review dataset. This dataset was released by Amazon in 2018 and it is available publicly. The dataset includes reviews (ratings, text, helpfulness votes) and product metadata (descriptions, category information, price, brand, and image features). The data is divided into different subcategories such as electronics / beauty products / software etc. We are planning to experiment with different subcategories and choose the best fit. For now we are working on the Software and the Fashion products subcategory.

The **reviews dataset** is used to get the reviews given by each user to a specific item. The **product and metadata** is used to map the items in the review dataset to their corresponding metadata.

The datasets are in a zip file which we convert to a json format before storing it into a Pandas Dataframe.

Fig:- View of the data frame containing the reviews from the review dataset.

	overall	verified	reviewTime	reviewerID	asin	style	reviewerName	reviewText	summary	unixReviewTime	vote	image
5	3.0	True	05 6, 2015	A3W11493KS6Z2L	B000K2PJ4K	{'Size:': ' Little Boys', 'Color:': ' White/Bl...	NaeNae	Waaay too small. Will use for futur children!	Oops!	1430870400	NaN	NaN
6	5.0	True	05 6, 2015	A3W11493KS6Z2L	B000K2PJ4K	{'Size:': ' Little Boys', 'Color:': ' Blue/Ora...	NaeNae	Stays vibrant after many washes	Great	1430870400	NaN	NaN
7	5.0	True	05 6, 2015	A3W11493KS6Z2L	B000K2PJ4K	{'Size:': ' Little Boys', 'Color:': ' Blue (37...	NaeNae	Stays vibrant after many washes	Good	1430870400	NaN	NaN
8	5.0	True	05 6, 2015	A3W11493KS6Z2L	B000K2PJ4K	{'Size:': ' Little Boys', 'Color:': ' Blue/Pink'}	NaeNae	My son really likes the pink. Ones which I was...	Great	1430870400	NaN	NaN

We later work on this dataframe on the basis of each one of our models.

Progress till now:

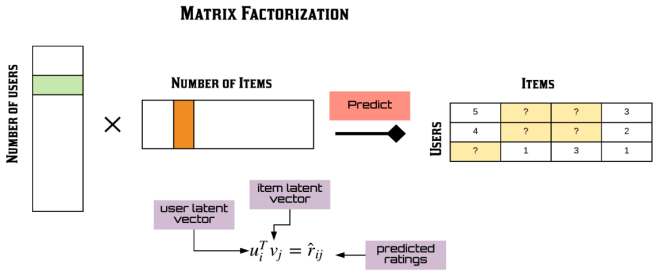
1. For **KNN**, from the dataset we kept reviewerID, asin (ProductID) and ratings columns and ignored the remaining. Further we have converted this tabular data into matrix format, which has product ids as rows and user ids as columns, that can be utilised by KNN. As we'll be working on distance calculating between vectors, we have filled missing ratings with zeros in this matrix.

2. For the **Deep learning Model NCF**, the data is split into train-test based on the timestamp. Now, instead of taking the actual ratings, we have considered the implicit ratings as in real life scenario most of

the users don't give ratings. So, we have converted the explicit ratings into implicit by considering the items rated by the user as 1 and considering 5 other items not rated by the user as 0. The value 5 is taken in general, we can change it and modify it, it is just to introduce the negative cases in our scenario. Thus there will be two final labels: 1 and 0. 1 will be if the user has interacted with that item and 0 if he has not. A baseline pytorch class is created for the dataset loader.

3. For our **Matrix Factorization algorithm** (SGD and ALS), we require 3 columns of data. We create a Utility Matrix with Users as rows, Items as Columns and ratings as the values, using these 3 columns from the original dataset. To decompose the User-Item matrix, the usual technique is SVD. However, SVD does not work with missing values. We used SVD only to initialise the P and Q matrices.

Fig:- Matrix Factorization



Cite: https://towardsdatascience.com/

After that, we will try to find the missing ratings by finding P and Q, such that P.Q(T) will give a close approximation for every known value of M. To minimize P and Q over the known values of M, we need to minimize our cost Function.

4. We Performed ALS and SGD based Matrix Factorization on 5-core Amazon Reviews datasets (subsets) - datasets containing items of 1 category . We have a pipeline in place to run the training set with various choices for the hyperparameters . RMSE is the evaluation function used to calculate the errors.

5. SGD is performing better on these datasets - much better learning curve for all hyperparameter choices.Training time is significant for moderately sized datasets.

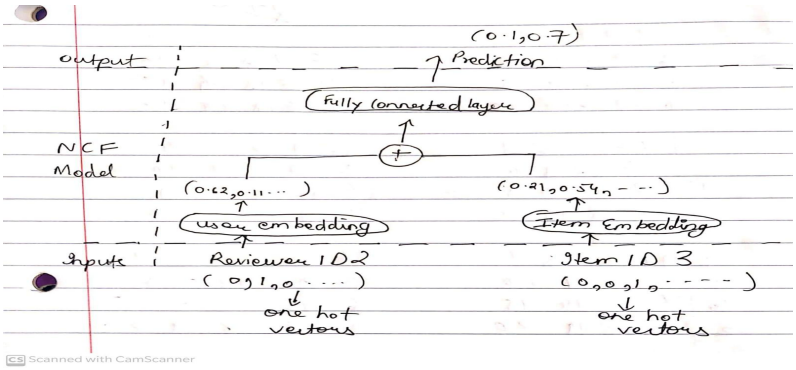
Upcoming Milestones:

- **Neural Collaborative Filtering:** One of the deep learning based models that we are going to try is the Neural Collaborative Filtering. Which has input as the user one hot vector and item one hot vector. The inputs will be passed to User Embeddings and Item Embeddings respectively. The embeddings help us categorize similar users and items based on the dimension space that we are using. Initially, we will start with 7 dimensions and might try out more depending on the feasibility. Finally the user and item embeddings will be concatenated and fed to the fully connected deep neural network. The output will be a probability of two classes i.e 0,1. For the model evaluation of recommender systems, traditional accuracy and RMSE are not appropriate.

The key here is that we don't need the user to interact with every single item in the list of recommendations. Instead, we just need the user to interact with at least one item on the list — as long as

the user does that, the recommendations have worked. A metric called as the hit-ratio will be used to evaluate the performance. Hit ratio determines if the actual interacted item lies in the predicted set or not.

Fig: Model Architecture



- For **KNN**, to understand the taste of users we'll only consider users who have rated more than 50 products and will only consider products which are rated more than 50 times. To calculate the similarity between products we'll try different distance metrics like Euclidean, Manhattan and Cosine similarity We'll train the model with different values of K to get the optimum results.For evaluating the model we will be using a 'hit ratio' metric.

Challenges ahead :

1. We have made progress on the problem of Matrix Completion(ratings prediction) of the User-Item ratings matrix, where using the above techniques we are able to get good performance on our test set.

2. We see slow convergence for even moderately sized datasets. We need to find a way to parallelize the solution or use more compute power. We will also need to document the training time.

3. We want to run our solution on larger subset datasets in the Amazon Reviews Datasets where many more categories of products are present to recommend products across categories.

4. The cold start problem of new users/ new items needs to be addressed - features of new users will not be present in our decomposed matrix, so this problem needs to be addressed using some other techniques. We will try to find a way of recommending existing and new users, top-N recommendations once the prediction matrix is generated from the training steps.

Code :

The code for our work till now is pushed into the git repository Here.