Recommendation Engine using Collaborative Filtering & product propensity estimation

Jubin Mohanty
Advance Analytics
Mashreq
Dubai, UAE
Email: mohantyjubin2211@gmail.com

***Abstract*— One of the critical problems that our legacy recommendation system was facing was that it could not accurately locate cross-opportunities and made vague product recommendations for both old and new customers. Because of this loophole in the context-based recommendation engine, I have proposed a multi-task recommender model using collaborative filtering. The model is integrated with a data pipeline, which can refresh customer data based on timestamps. We have also set up a customer sentiment model to understand customer behavior towards our products from existing call reports. The architecture is developed using a TensorFlow deep neural network recommender. In order to obtain the recommendation, I performed two tasks which are retrieval and ranking. A query tower and candidate tower have been set up to perform retrieval, which contains stacked embedding layers of the users, product names, and other candidate features. The embedding output query-affinity score will express the match between query and candidate. Additionally, to rank our top five recommended products, I have developed a Relu-based deep neural network. To evaluate the recommendation model, I have proposed an offline testing approach as compared to online testing. This testing pipeline has evaluated the model based on hit rate, novelty, coverage, and diversity. For the product propensity, we calculated the threshold using logistic regression, which further explains customer eligibility for the product.**

I. INTRODUCTION

As the web has become one of the mediums for business transactions, and that served the development of recommendation systems. The E-commerce and entertainment industries have gained tremendous success in boosting their product sales. Also, for the Banking sector, a personalized product recommendation system is the key to achieving their desired business goals[2]. Bank has a variety of products that are bifurcated into two domains, which are corporate banking and retail banking. Digital products and services such as debit cards, credit cards, fund transfers, internet banking, saving accounts, fixed deposits, treasury bills, student loans, and mortgage loans are typical examples of the retail banking Industry. Whereas corporate deals with large conglomerates with billions in sales. The commercial bank offers loan and credit products, treasury and cash management services, Equipment Lending, Trade finance, and commercial real estate. Therefore, it became imperative for banks to sell products to the right customers and provide recommendations to the existing as well as prospective customers.

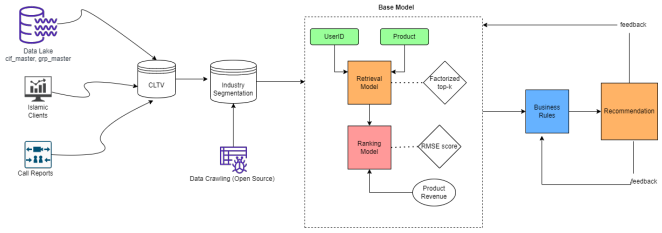


Fig. 1. Recommendation Engine Architecture

Even after the inclusion of various recommendation technologies such as collaborative filtering, content-based filtering, knowledge-based recommendation, and hybrid strategies, there are hardly any tailored for banking applications[3].

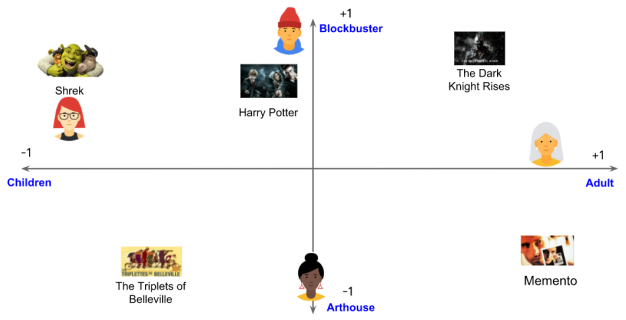
This paper will discuss a hybrid approach where we used customer segmentation based on their business, model-based collaborative filtering, and a relu-based deep neural network to rank the products.

II. BASICS

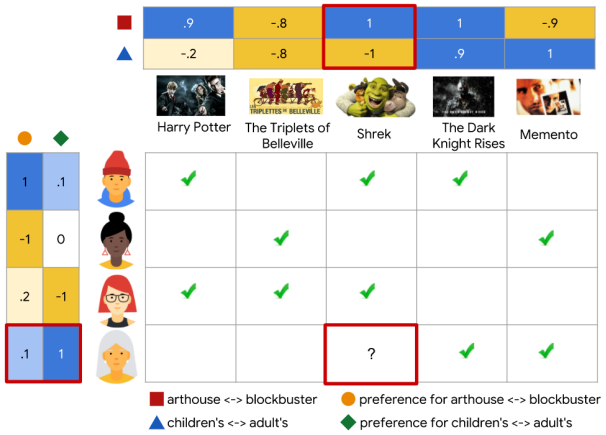
In our business problem, we were looking to get cross-opportunities for the customers, such that we can explore product recommendations for users outside their business domain based on their candidacy with other clients. Because of this reason, we went with a collaborative Filtering methodology, which can automatically learn with highly aligned hand-engineered engines.

Let's consider a movie recommendation system, in which the training data consists of explicit(particular movie rating), and implicit(interested users) data. Here, our feedback matrix will be binary, where 1 indicates positive sentiment. To develop a recommendation model which incorporates, the move similarity that user liked in the past and movies which belongs to similar user, we can build user item embedding ranging from 1-D to n-D.[6]

For instance, let assume we are going to recommend movies for children and adults, where we have assigned a scalar value of [-1,1] both to each movie and a user. Also, the negative values are for children and positives for adults. Additionally, if we want to understand the user preference



(a) 2D Embedding



(b) Feedback Matrix

Fig. 2. Matrix Factorization; (user,item)pair

more profoundly lets add one more feature which explains the movie to be a blockbuster or Arthouse.

A matrix is an embedding model which learned in such a way that the dot product of $U.V^T$ provides an accurate approximation to the feedback matrix.[1]

For a given matrix $A \in R^{m*n}$, where n denotes the number of items and m as the number of users.

- User embedding matrix; $U \in R^{m*d}$
- Item embedding matrix; $V \in R^{m*d}$

III. Deep Neural Network Model

A. Model Architecture

One of our requirement is to use additional side features in the embedding layer beyond query and item, such that we can capture user specific interest and further improve our recommendations. As matrix factorization has this limitation, therefore, we decided to achieve collaborative filtering using DNN.

In DNN model, one possible way would be using softmax which treats the problem as multi-class prediction[10] where:

- Input is the user query which contains dense features(for example time), and sparse feature such as history, country.
- The output is a probability vector containing items equal to the size of corpus, represents the probability to interact with each item. For instance, the probability to watch or click a video on YouTube

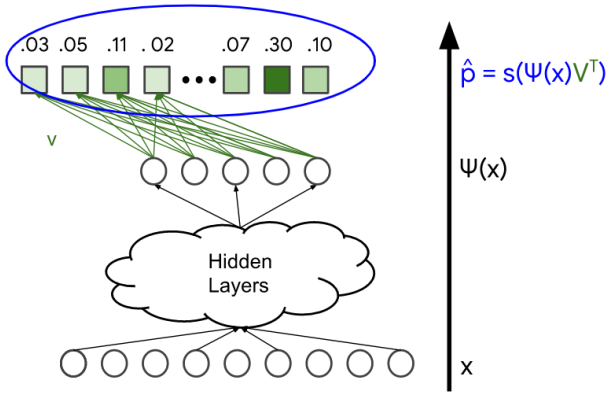


Fig. 3. The predicted probability distribution, $\hat{p}$ = h($\psi(x)V^T$)

Looking at our use case, we went with a two-tower neural network in which the model learn a non-linear function that maps features to an embedding.

This architecture flexible enough to handle hidden layers and activation functions such as RELU, which enables the model to capture more complex relationship.[11]

The last hidden layer is denoted as $\psi(x) \in R^d$ and the probability distribution of the softmax layer is $\hat{p} = h(\psi(x)V^T)$,where:

- The softmax function $h : R^n \to R^n$, given by $h(y)_i = \frac{e^y}{\Sigma_j e^y j}$
- The weights of the softmax layer matrix is $V \in R^{n*d}$

The final model output can be defined as the dot product of $< \psi(x_{querry}), \phi(x_{item}) >$, and the model predicts one value per pair (x_{querry}, x_{item}).

IV. Hybrid Model

A. Dataset

We have used our corporate customer database from which we have curated their revenues across the products, credit ratings, and annual turnovers. For their respective business activities, we crawled data from various open web sources and, using fuzzy matching, clustered it together based on similarity for each user.

In order to update our legacy database of customer product lists, we developed an entity recognition model to automatically identify customers and products from news articles using bi-directional LSTM and conditional rational fields.[9]

B. Training Pipeline

- **Data Curation:** After curating customer revenues, sentiments, product information, turnovers, and business information, the data frame is converted into TF dictionary.
- **Create Identifiers:** I have created identifiers named interaction objects which contain the dictionary of all the features.

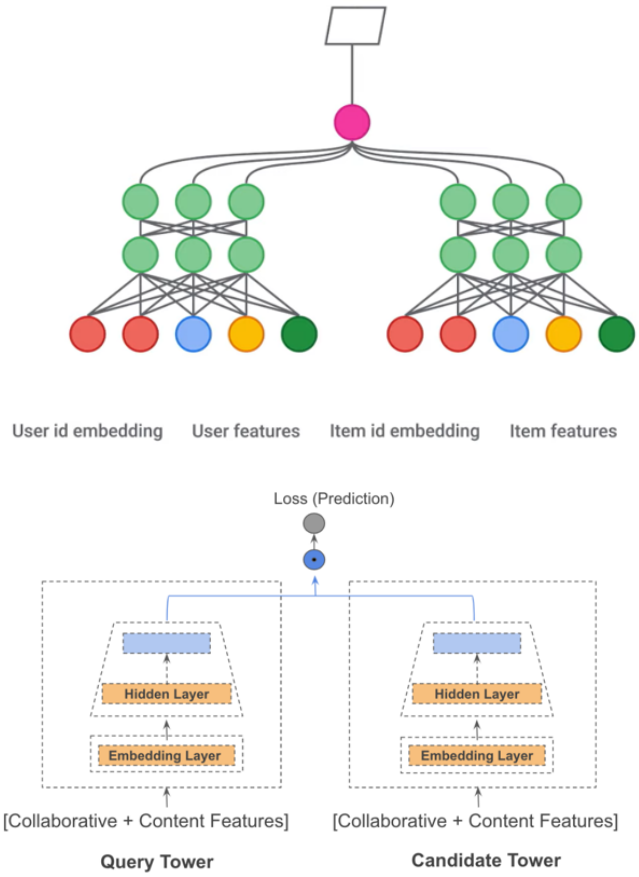
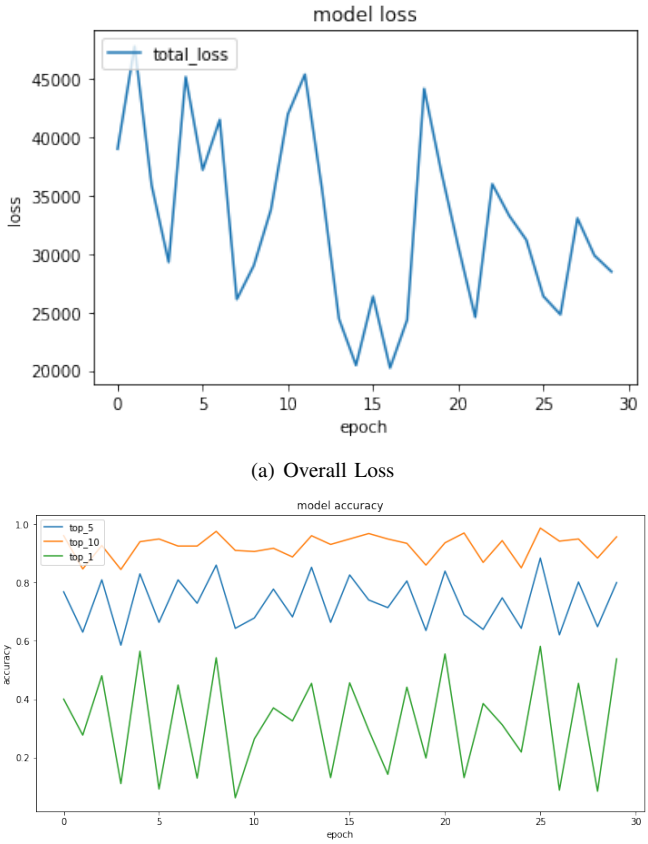


Fig. 4. Two tower neural network model



(a) Overall Loss

(b) Model accuracy

Fig. 5. Model Training and Validation

- **Build Lookup Table:** A lookup table, in other words, a reference map is created to get categorical and numerical features as an embedding vector in a retrieval model.
- **Data Prepossessing:** Entire data has been shuffled and split in 80-20 ratio as train and test segments.
- **Retrieval Task:** Here, I have defined the sequential layers and the model tower. The model tower contains a candidate(product) model, query(customer) model, retrieval task to select top-5 products using *factorized top-k* metrics, and finally, a loss computation component.[5]
- **Ranking Task:** The steps performed here are similar to that of Retrieval, the only difference is with the usage of accuracy metrics RMSE instead of factorized top-k. [8]
- **Propensity:** To know the product propensity for each customer, I developed a *logistic regression* model to estimate the probability of product importance for each customer. It further helps us to weed out the product which has less importance. [7]

V. Testing

For evaluating our model, we initially went with online testing, as we are looking for a quick and cheap way to measure the performance of the model. Our baseline model is a simple retrieval recommendation engine that is trained without multi-feature embedding layers.

The metrics used are Hit Rate, Coverage, and Novelty.

HitRate	Coverage	Novelity
41.3%	60%	18%

TABLE I
BASE RETRIEVAL MODEL

HitRate	Coverage	Novelity
61.53%	99%	33.13%

TABLE II
MULTI-FEATURE HYBRID MODEL

We have leveraged the Novelty principle to identify the popularity of the products. It takes the average of all the popularity rankings for products predicted to the customer.[4]

VI. Conclusion

We found out that our Hybrid model, consisting of the collaborative filtering deep learning model and rule-based feed-

back loop, outperformed the context-based filtering model. Our legacy context-based filtering model depended highly on hand engineering data and product propensity against each customer. Also, as it becomes a classification problem, thus maintaining each classification model for a product became highly difficult and costly. Further, along with offline testing, our online testing is in progress as it can truly evaluate the model performance. Additionally, our placeholder for further improvement is Novelty. Including more features that should not show collinearity and can uniquely segregate each customer can increase the rate of predicting new cross-domain products and improve the Novelty score. Also, we are constantly negating the False positives from the customer business information and product data, which we have identified using the Flair-based entity recognition model from open source articles.

REFERENCES

[1] Matrix factorization. `https://developers.google.com/machine-learning/recommendation/collaborative/matrix`, Feb 2029. Google Developers.

[2] Charu C Aggarwal. An introduction to recommender systems. In *Recommender systems*, pages 1–28. Springer, 2016.

[3] Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

[4] Jun An, Songzheng Zhao, Xiaoni Lu, and Ningning Liu. A two-stage multiple-factor aware method for travel product recommendation. *Multimedia Tools and Applications*, 77(21):28991–29012, 2018.

[5] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[6] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[7] Adrian Mackenzie. Personalization and probabilities: Impersonal propensities in online grocery shopping. *Big Data & Society*, 5(1):2053951718778310, 2018.

[8] Nour Nassar, Assef Jafar, and Yasser Rahhal. A novel deep multi-criteria collaborative filtering model for recommendation system. *Knowledge-Based Systems*, 187:104811, 2020.

[9] Rrubaa Panchendrarajan and Aravindh Amaresan. Bidirectional lstm-crf for named entity recognition. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, 2018.

[10] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pages 1–7. 2017.

[11] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, 69:29–39, 2017.