Collaborative Filtering based Recommending systems with Tensorflow

Trupthi Hosahalli Chandrappa

***Abstract*— Online or retail world has been benefit-ted with the many recommender systems that has been beneficial for both the service providers and consumers who are looking for browsing, learning or buying things online. I, well I confess, have fallen for and sometimes indulged myself in some of the suggestions I have gotten when browsing through the internet shopping websites where I am suggested with items to buy. I always wondered how does the company know what do I want and they would realize it or identify the needs of a buyer.**

As I scouted to get a good understanding of this suggestive systems while learning the CS410 course I picked appropriately led me to the understanding of how things worked in the background; how in a way I was collaboratively contributing their systems. With a good foundation set, I ventured into the realm of exploring Collaborative Filtering based Recommender systems.

I. INTRODUCTION

Recommender systems in general are machine learning systems that help users and eventually providers to help users discover new products and services. These systems are intelligently nudging the users towards the most plausible product or service they might be interested in purchasing or availing oneself of. These systems have been integral part of the digital world helping users to consume the enormous data content or online to make choices in a better organized way in finding items that they are looking for, rather getting overwhelmed with tons of information on the internet. These systems work based on history and preferences of the users, collecting feedback from similar users making it easier to distinguish relevancy of products or services.

II. FUNCTIONAL AREA OF RECOMMENDER SYSTEMS

Recommender systems have now been an integral part of many online websites providing enhanced user experiences and also unveiling us to inventories that we may not otherwise discover ourselves.

These systems work based off of relationships between user-product, product-product and user-user relations. They also collect data on user behavior, user location/demographic and feature of products. Data for such decision making is done based on Explicit ratings (provided by the users) and Implicit(based on interaction with the item) ratings.

These decision making based on some of the similarity measures that are derived based Product similarity or User-user similarity. These metrics are measured using the distance matrix. Some of these measuring include Minkowski Distance (dimension of the data point is numeric), Manhattan distance (distance measured along axes at right angles), Euclidean Distance (L2 Norm), Cosine Similarity (measures cosine angle between two vectors), Pearson co-efficients (measure of correlation between two random variables in the range [-1,1]), Jaccard Similarity (similarity between two finite sets), Hamming Distance (categorical variable based distance calculation). [3]

A. Types of Recommender Systems

There are few strategies that these systems appertain to improvise suggestions to users. These work based on feedback information collected from multiple like minded users who have previously engaged with the system, helping future prospective or returned users to make informed and relevant choices based on prior engagements. There are two kinds, Content-Based Filtering and Collaborative based filtering to help users make decisions with more relevance.

Content-Based filtering recommenders work well when descriptive data on the content is provided and similarity is measured based on product attributes. This information can be obtained based on rated content and description of the content.

Collaborative filtering recommenders works based on users rating in the past and not based on the product itself and similarity measures are measured against the similarity of the users.

Systems can also be built on hybrid models which can be a combination of Content-based and Collaborative-based filtering.

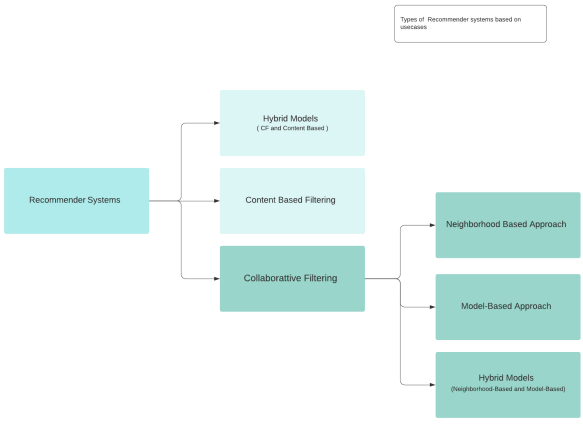

Fig. 1. Types of Recommender Systems

This review explores more on the Collaborative Recommending systems, it's functionality, implementations and use-cases.

B. Collaborative Recommending Systems

These systems work based off of similar users' will have similar preferences and such users share similar interests.

They utilize users' rating and predicts or filters products or services based on these ratings/data. Basic idea here is predict the rating of a current user for a product based on the weighted average of the ratings of the product given by similar users as the current active one. [6] It can be perceived to have two steps, first to procure similar users to the current user (defined as the similarity between the two vectors for two users) rating vector, where Similarity can be measured on cosine similarity or Pearson Correlation of the two vectors and second weighted average of the rating is calculated which is the correlation between the active and the corresponding similar user. Ratings can take many forms like Scalar, Binary and Unary.[2] This filtering can be viewed as a user x product matrix with moderately filled indices, that represents user feedback. Key point to note here is the collaborative recommender systems feed off of the relationship between the user and product. Hence making it a generic or more prevalent application to any item/product/service. Getting to know this information with at-most sufficiency to make accurate predictions may not always be available due to data sparsity, called as **"Cold Start Problem"**.

III. Uses of Collaborative Filtering

It will be quiet interesting to explore the use-case for choosing collaborative filtering approach over other strategies. Designers of web applications often carefully curate and identify several user tasks that would encourage users in making their decisions quicker and avoid overwhelming them with loads of information. This paper details some great use-cases that are considered when building such web applications which internally helps marketing[2]. These systems thrive in conditions where data distribution on the products or services are vast and colossal; also there are multiple users already contributing to this data, for other users to be truly benefit-ted in making those choices with better predictions. Such systems bring value when its dealing with items having similar underlying meaning, following which it also needs to ensure the relevance of such data collected or obtained. Collaborative filtering works well with active users' participation and its rating. Developing a recommending system involves many strides like data acquisition, building the vectors, establishing the model, evaluating the model and defining the course of action for coding and deploying. With advancement in research and technology, we now have simplified open-source packages that reduces the complexity of developing a recommender systems. One such implementation is a library provided by Tensorflow, called Tensorflow Recommenders.

IV. TensorFlow Recommenders

TensorFlow Recommenders (TFRS) is an open-source library that simplifies building recommender system models, by helping us to construct the complete workflow, involving data preparation, model formulation, training, evaluation and deployment. This is based on TensorFlow 2.x and Keras and aims to providing the flexibility to build and evaluate complex models, easily incorporate item, user and context information and help train multi-task models and optimize multiple recommendation objectives; making it instantly familiar and user-friendly. It is modular by design, easily customizable individual layers and metrics, having a cohesive whole such that components work well together.

A. Fundamental Understanding

TRFS is an open-source library that makes structuring of recommender system models easy, fast, scalable. TRFS leverages ScaNN, a state-of-the-art NNS (nearest neighbor search) library from Google Research, which focuses on the compression of dataset vectors enabling fast approximation of distance computation, boosting accuracy, it was recently open-sourced as vector similarity search library; enabling retrieval of best candidates out of millions in milliseconds. The key point here is the importance to simplicity. Sparsity, quantization, and architecture optimization are key techniques to consider to help reducing cost of retrieval. TRFS also includes better or enhanced techniques modeling feature interactions, with effective feature crosses (combination of features of products). With TensorFlow recommenders, incorporating DCN (Deep and Cross Network)[1] which is designed to be smart in learning bounded-degree cross features more effectively; made up of three tiers first with an input/embedding layer and followed by cross network that models explicit feature interactions and finally a deep network that models implicit feature interactions. For better model understand-ability, having a good understanding of the learned feature crosses helps. Cross layers are implemented in TensorFlow recommenders so it can be easily adopted in model building. TFRS puts together an concoction of interesting qualities like:

- Constructing and assessing flexible candidate nomination models
- Simplifies incorporating item, user, and context information into recommendation models
- Trains multi-task models which jointly optimizes multiple recommendation objectives
- Efficiently serves the resulting models using TensorFlow Serving.

Throughout the design of TFRS, emphasis is on the flexibility and ease-of-use: with settings sensible and common tasks intuitive and straightforward to implement; more complex or custom recommendation tasks is easily implementable.

B. Sample Implementation

For a newbie, like me, I wanted to explore the various aspects a recommender systems needs for building one with TRFS. It involves TRFS set-up or import, read the data, define a model and finally fit & evaluate the model [5]

Similar to the tutorial, I worked with the notebook with the movie lens dataset to learn about the different aspects of implementing a recommender system.

- Import TRFS : Set-up is pretty straightforward, with a pip install the library can be installed. pip install

-q tensorflow-recommenders pip install -q –upgrade tensorflow-datasets pip install scann
import numpy, tensorflow as tf, tensorflow_datasets as tfds and tensorflow_recommenders as tfrs

- Read the data : Sample dataset that I explored was the dataset from movielens, with ratings from 100k users.

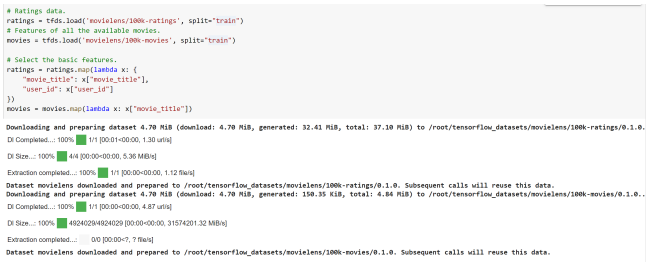

Fig. 2. Reading Movie-lens Data

For the embedding layer, we will be building the vocabularies with user id and movie titles

- Define a model : TRFS model can be built by inheriting trfs.Model and implementing a method for building the embeddings

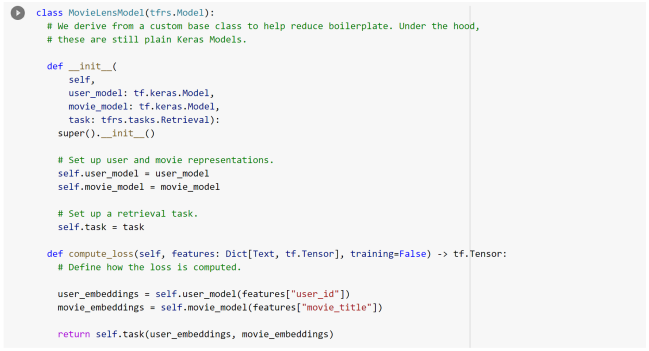

Fig. 3. Define Model

Build user and movie models with user_id vocabularies and utilize the embeddings with tf.keras.Sequential Also, define the objectives with tf.keras.Retrieval and evaluation metrics

- Fit and evaluate the model Finally, create the model, train and evaluate. Tried both BruteForce and ScaNN approach

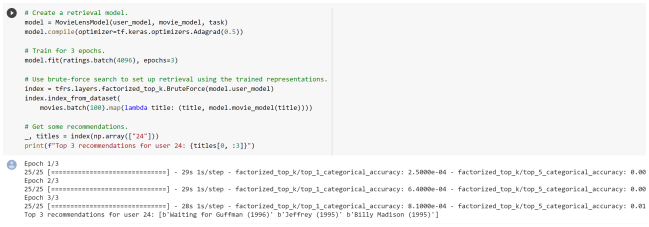

Fig. 4. BruteForce based Model

Interesting point to note was that the performance was not too drastically different since the dataset was pretty

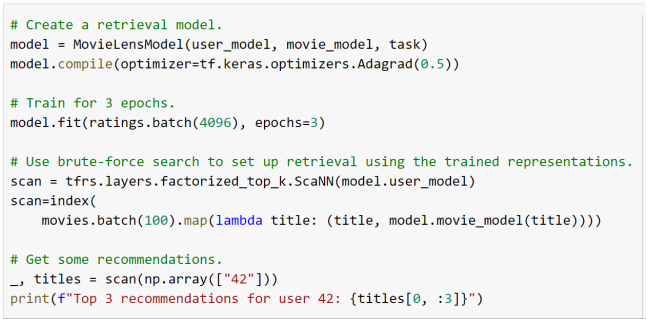

Fig. 5. ScaNN based Model

small, but ScaNN will should performance improvements when the dataset is huge or in millions[4]

V. CONCLUSIONS

For someone who is learning the concepts of NLP processing and the multi-faceted use-cases and implementations, learning about recommendation systems gave me an opportunity to learn and help me understand some of these underlying implementations and various use-cases that are currently being extensively used by the many online shopping and other retail websites. Things that intrigued me as magic, now has a logical explanation on the things that go in the background for these applications work so well in real world.

.

REFERENCES

[1] “Deep & Cross Network (DCN)”. In: (). URL: https://www.tensorflow.org/recommenders/examples/dcn.

[2] Schafer J.B. Frankowski D. Herlocker J. and Sen S. “Collaborative Filtering Recommender Systems”. In: (2007). DOI: https://doi.org/10.1007/978-3-540-72079-9_9.

[3] Badreesh Shetty. “An In-Depth Guide to How Recommender Systems Work”. In: (2019, Updated 2021).

[4] “TensorFlow Recommenders: Efficient Retrieval”. In: (). URL: https://colab.research.google.com/github/tensorflow/recommenders/blob/main/docs/examples/efficient_serving.ipynb#scrollTo=qRI-qv7S2h97.

[5] “TensorFlow Recommenders: Quickstart”. In: (). URL: https://www.tensorflow.org/recommenders/examples/quickstart.

[6] ChengXiang Zhai and Sean Massung. “Text data management and analysis: a practical introduction to information retrieval and text mining, Chapter 11”. In: (2016).